Predicting VIX: A Time Series Analysis

Nithin Premkumar 1005555436

Table of contents

Abstract

The availability of a good predictor of implied volatility is a desirable knowledge that a person or a firm would need for participating in the stock exchange market. Time Series Analysis is a specific way of analyzing a sequence of data points collected over time [5]. With the availability of information on past VIX values, we will apply time series methods to analyze VIX and predict its behavior in the future.

In our analysis, we discover that it is possible to forecast future values of VIX by training a model on the present and past values. The predictions made from this project should be taken with a grain of salt, as there are factors that can influence VIX. These external influences are tough to code in our model or predict by the model. It is better to train our time series model on data relevant to the present, as different periods often have a different trend of VIX.

Introduction

Market volatility is one of the critical factors that determine a nation's economy at the moment; in an open economy, many factors can affect this. Therefore it is crucial to understand realized and implied volatility; realized volatility assesses variation in returns for an investment product by analyzing its historical returns within a defined period [1]; implied volatility is the market's forecast of a likely movement in a security's price. It is a metric used by investors to estimate future fluctuations (volatility) of a security's price based on certain predictive factors. The security's price reflects the value of the asset underlying it [2].

This project deals with understanding implied volatility and trying to develop a model that can forecast future fluctuations. In order to accomplish this task, we will be using Monthly CBOE Volatility Index (VIX). VIX is a real-time index representing the market's expectations for the relative strength of the S&P 500 index (SPX). The data set contains information from 1990, February 01 to 2021, December 13. [3]

```
##         Date   VIX
## 1 1990-02-01 21.99
## 2 1990-03-01 19.73
## 3 1990-04-01 19.52
```

```
##           Date   VIX
```

```
## 382 2021-11-01 27.19
## 383 2021-12-01 19.90
## 384 2021-12-13 19.20
```

In order to make predictions, we should be somewhat skeptical of our results, and this is because specific events, global or local, can influence VIX. These shocks are unpredictable and can significantly affect the volatility causing a drastic change in VIX.

The data trend is not something constant the trend in the 1990s may no longer be applicable for making predictions for future values. Therefore, while training our model, we should be cautious of this. Nonetheless, forecasts help gauge a possible future value of VIX.

A model is as good as the data it is trained upon, having relevant data is more critical than having extensive data. The following plot can further explain the relevancy of data for future values:

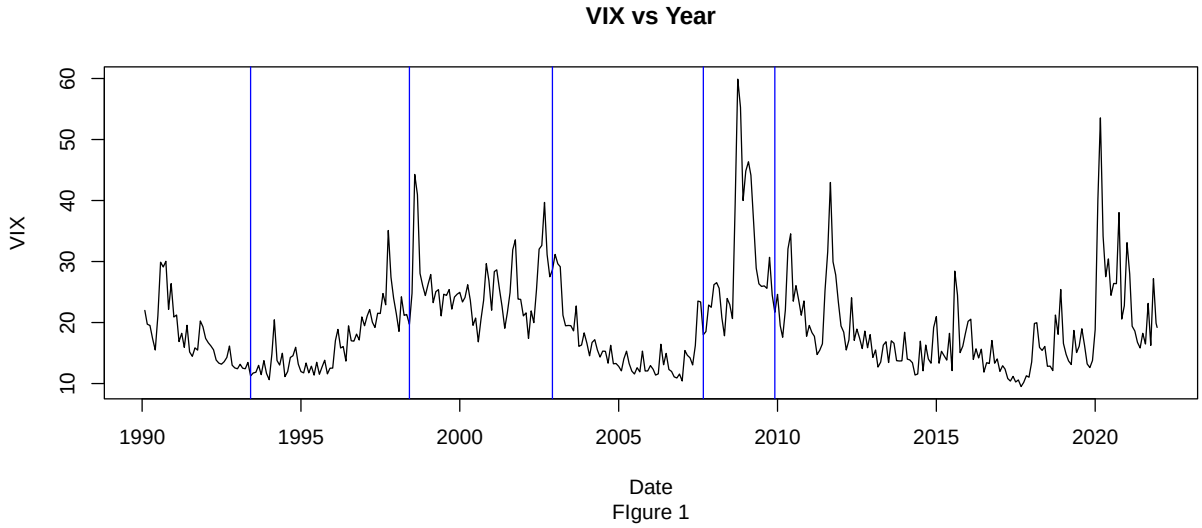


FIgure 1

The plot shows that we can form six segments across time according to the general trend of the VIX. However, we can see some extreme peaks in some sections that deviate from the general trend. Possible causes to these spikes could be:

Section 1: the major incident was Gulf War 1, an armed campaign waged by United States-led coalition of 35 nations against Iraq in response to the Iraqi invasion and annexation of Kuwait. 1990, August – 1991, February.
Section 2: had no drastic peaks.
Section 3: The Russian financial crisis hit Russia on 17 August 1998. It resulted in the Russian government and the Russian Central Bank devaluing the ruble and defaulting on its debt. In 2001, stock prices took a sharp downturn in stock markets across the United States, Canada, Asia, and Europe.
Section 4: no significant spikes.
Section 5: The bankruptcy of Lehman Brothers in 2008 September was the climax of the subprime mortgage crisis.
Section 6: the debit-ceiling crisis of 2011 due to the massive increases in federal spending following the Great Recession of 2008. The covid-19 pandemic hit on March 2020 worldwide.
This project will take two perspectives on this matter, one model will consider the entire dataset for training, and the second model will consider the data from 2010 January only.

Model Specification

In order to forecast future VIX data, a model has to be developed that replicates a general behavior of VIX; this can be accomplished by developing an Autoregressive Integrated Moving Average (ARIMA) model. Hence, these models have assumed great importance in real-world modeling processes. An ARIMA model has three parameters, p - order of autoregressive part, d - differencing order, and q - order of moving average part. The challenging part of this assignment is to find optimal values for these parameters.

There are general workflows to figure out the values of parameters; this involves plotting the data set, mean-level plot, plotting the autocorrelation function (ACF), and the partial autocorrelation function (PACF).

Model 1: Considering the entire dataset

Part of the data is excluded from training the model to test its validity. Therefore 10% of the data is test data, and the remaining 90% is training data. The values of p,d, and q are determined using the training data.

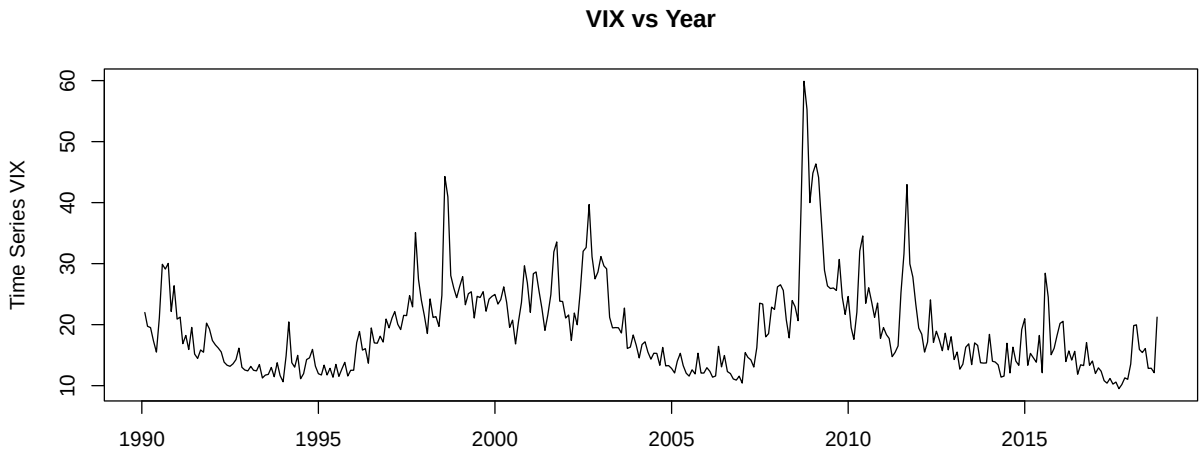


Figure 2

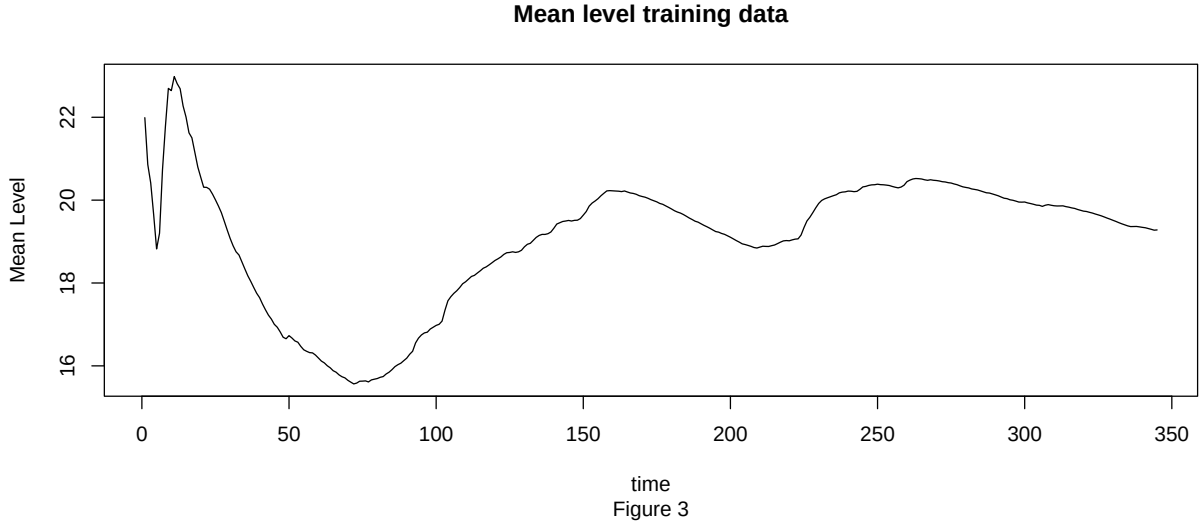


Figure 3

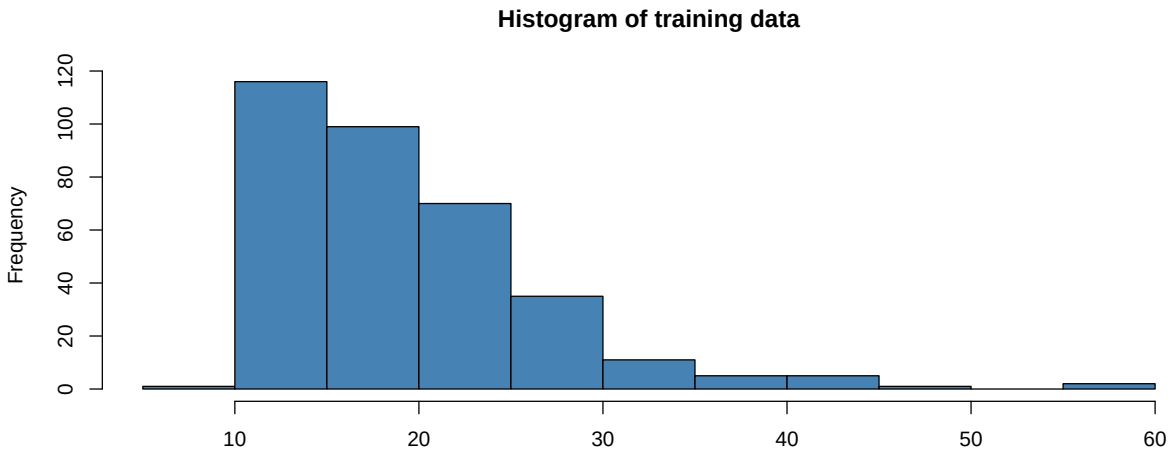


Figure 4

There is no stationarity in the training data from the time plot as VIX values rise and fall frequently; this is also seen in the mean level plot as some cliffs. The data also seemed too heavily skewed to the right from the histogram.

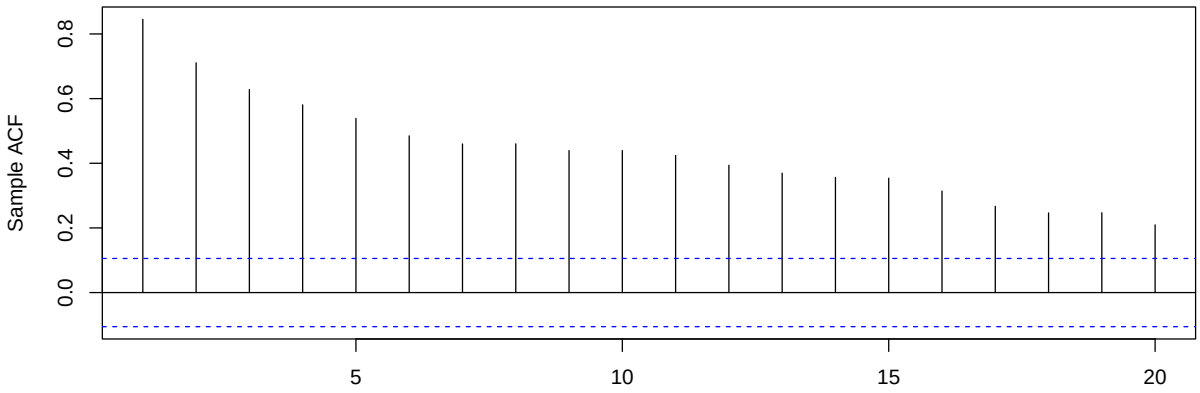


Figure 5

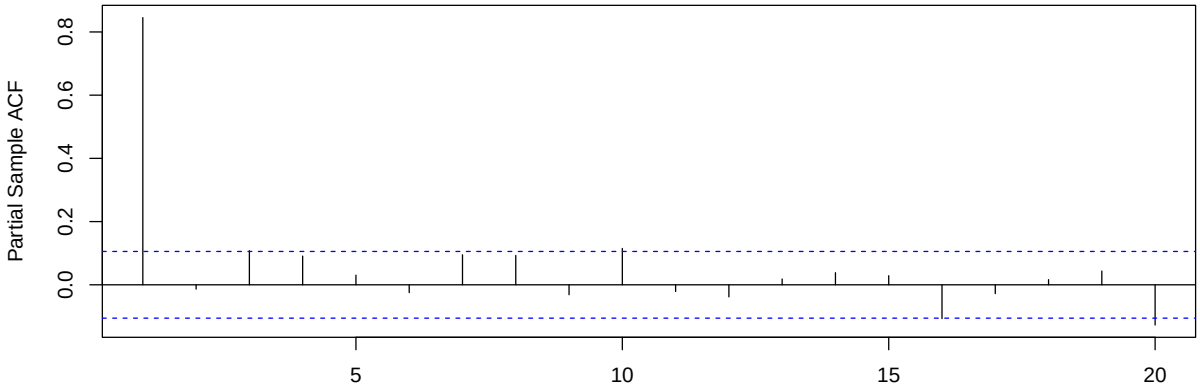


Figure 6

By analyzing the ACF and PACF plots, we can see that the ACF plot does not tend to 0 quickly, which is a sign of non-stationarity. An ADF and KPSS test can be carried out to get statistical inference about stationarity.
If the p-value of ADF test is greater 0.01, we can infer that data is non-stationary.
If the p-value of the KPSS test is lower than 0.01, we can infer the data is non-stationary.

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -1.88  0.0611
## [2,]   1 -1.75  0.0808
## [3,]   2 -1.48  0.1502
## [4,]   3 -1.27  0.2250
## [5,]   4 -1.16  0.2656
## [6,]   5 -1.20  0.2505
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -5.35    0.01
## [2,]   1 -5.21    0.01
## [3,]   2 -4.48    0.01
## [4,]   3 -3.96    0.01
## [5,]   4 -3.74    0.01
## [6,]   5 -3.74    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -5.35   0.010
## [2,]   1 -5.20   0.010
## [3,]   2 -4.47   0.010
## [4,]   3 -3.96   0.011
## [5,]   4 -3.74   0.022
## [6,]   5 -3.74   0.022
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag stat p.value
##    4 1.58  0.0581
## -----
##  Type 2: with drift no trend
##  lag  stat p.value
##    4 0.153     0.1
## -----
##  Type 1: with drift and trend
##  lag  stat p.value
##    4 0.152  0.0447
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

We can conclude that the data is non-stationary.

It is easier to analyze stationary time series data. Therefore, the training data were transformed by taking a difference of the inverse of the training data. The inverse was taken to normalize the training data, and an ADF and KPSS test was tested on the transformed data.

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -24.68    0.01
## [2,]   1 -16.94    0.01
## [3,]   2 -13.94    0.01
## [4,]   3 -12.31    0.01
## [5,]   4 -10.63    0.01
## [6,]   5  -9.82    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -24.65    0.01
## [2,]   1 -16.91    0.01
## [3,]   2 -13.92    0.01
## [4,]   3 -12.29    0.01
## [5,]   4 -10.61    0.01
## [6,]   5  -9.81    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -24.61    0.01
## [2,]   1 -16.88    0.01
## [3,]   2 -13.89    0.01
## [4,]   3 -12.27    0.01
## [5,]   4 -10.60    0.01
## [6,]   5  -9.79    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag   stat p.value
##    4 0.0306     0.1
## -----
##  Type 2: with drift no trend
##  lag   stat p.value
##    4 0.0271     0.1
## -----
##  Type 1: with drift and trend
##  lag   stat p.value
##    4 0.0276     0.1
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

We can conclude that the transformed data is stationary.

The ACF and PACF plots of the transformed data were taken to identify AR or MA signatures.

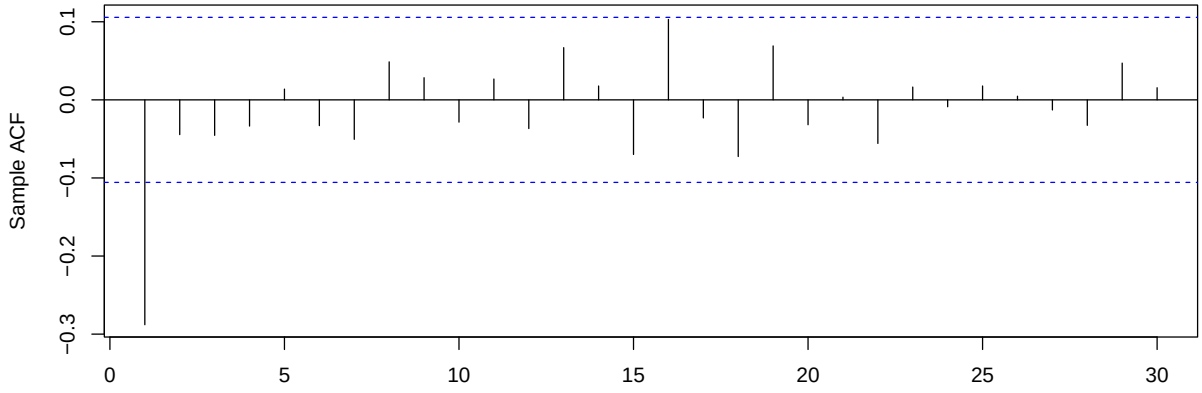


Figure 7

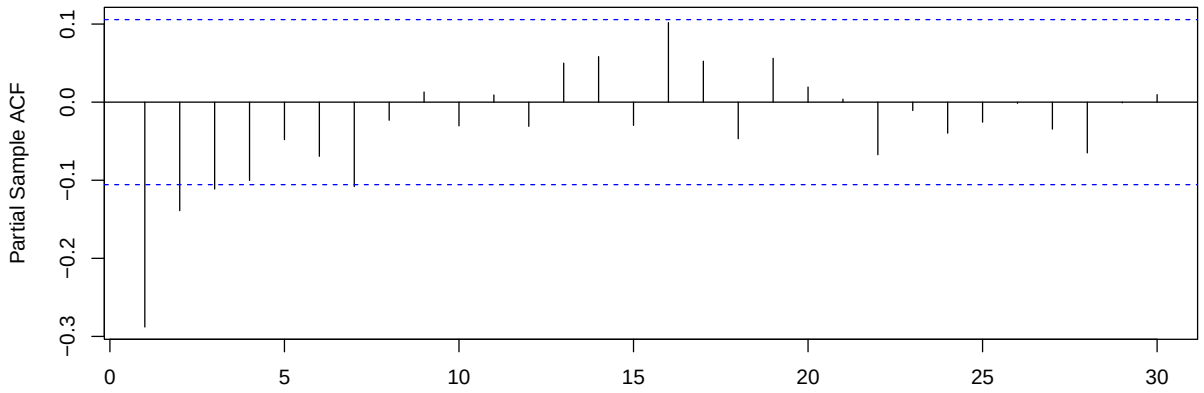


Figure 8

From the ACF plot, we can assume the MA order, q = 1.
From the PACF plot, we can assume the AR order, p = 2.
However, we can find the optimal value of p and q by finding orders with the lowest AIC and BIC scores.
After testing for this, p = 2 and q = 2. The above plots do not indicate any seasonality too.
Therefore, we choose these parameters as our first model's parameters.

Model 2: Using data since 2010 January

To test the model, we use data from 2021 January, and the remaining data is used for evaluating the values of p, d, and q.

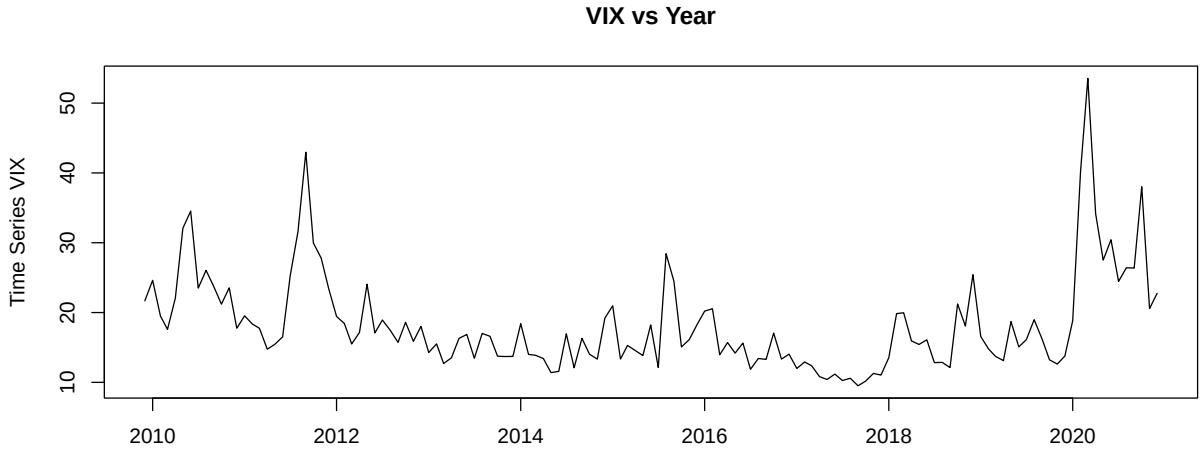


Figure 9

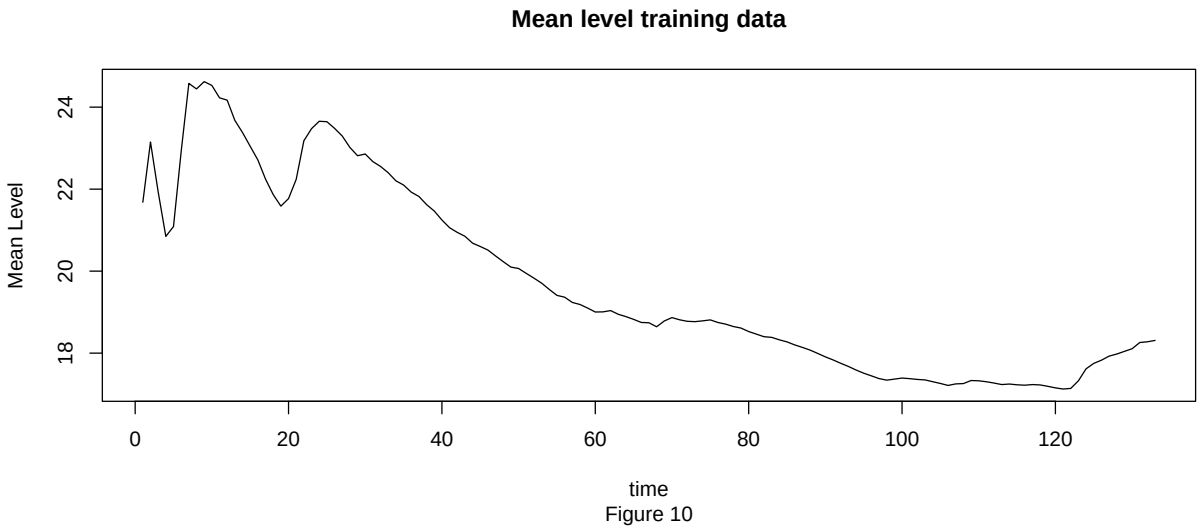


Figure 10

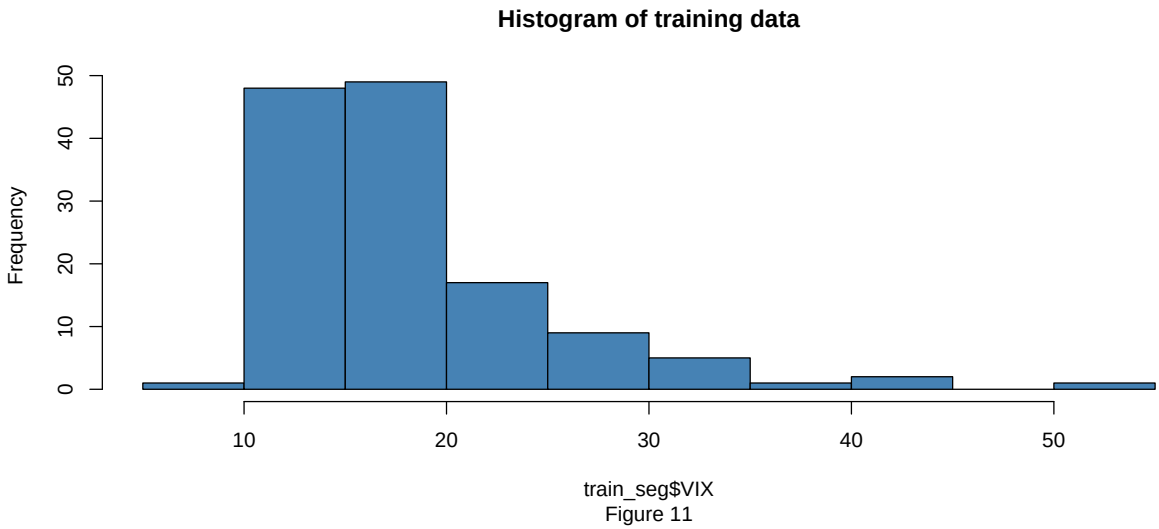


Figure 11

There is no stationarity in the training data from the time plot as VIX values rise and fall frequently; this is also seen in the mean level plot as some cliffs. The data also seemed too heavily skewed to the right from the histogram.

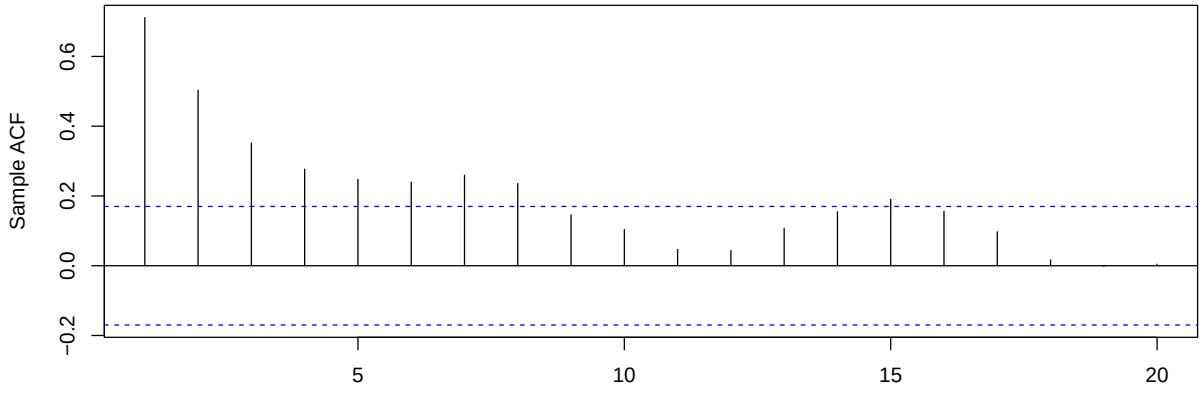


Figure 12

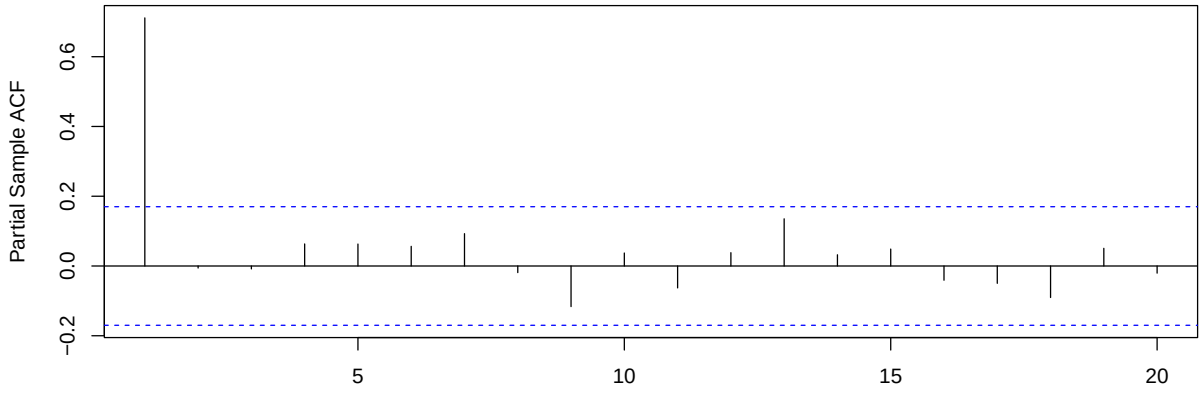


Figure 13

By analyzing the ACF and PACF plots, we can see that the ACF plot does not tend to 0 quickly, which is a sign of non-stationarity. An ADF and KPSS test can be carried out to get statistical inference about stationarity.

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -1.548   0.123
## [2,]   1 -1.406   0.174
## [3,]   2 -1.089   0.288
## [4,]   3 -0.843   0.377
## [5,]   4 -0.738   0.414
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -4.66  0.0100
```

```
## [2,]   1 -4.33  0.0100
## [3,]   2 -3.89  0.0100
## [4,]   3 -3.39  0.0145
## [5,]   4 -2.99  0.0409
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -4.63  0.0100
## [2,]   1 -4.30  0.0100
## [3,]   2 -3.86  0.0182
## [4,]   3 -3.36  0.0636
## [5,]   4 -2.94  0.1854
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag stat p.value
##    2  0.8     0.1
## -----
##  Type 2: with drift no trend
##  lag  stat p.value
##    2 0.193     0.1
## -----
##  Type 1: with drift and trend
##  lag  stat p.value
##    2 0.192   0.019
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

We can conclude that the data is non-stationary.

The training data was transformed by taking the difference of the training data, and an ADF and KPSS test was tested on the transformed data.

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -13.14    0.01
## [2,]   1  -9.71    0.01
## [3,]   2  -8.65    0.01
## [4,]   3  -7.94    0.01
## [5,]   4  -7.51    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -13.09    0.01
## [2,]   1  -9.68    0.01
## [3,]   2  -8.61    0.01
## [4,]   3  -7.91    0.01
## [5,]   4  -7.47    0.01
```

```
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -13.05    0.01
## [2,]   1  -9.64    0.01
## [3,]   2  -8.58    0.01
## [4,]   3  -7.90    0.01
## [5,]   4  -7.52    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag   stat p.value
##    2 0.0336     0.1
## -----
##  Type 2: with drift no trend
##  lag   stat p.value
##    2 0.0295     0.1
## -----
##  Type 1: with drift and trend
##  lag   stat p.value
##    2 0.0137     0.1
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

We can conclude that the transformed data is stationary.

The ACF and PACF plots of the transformed data were taken to identify AR or MA signatures.

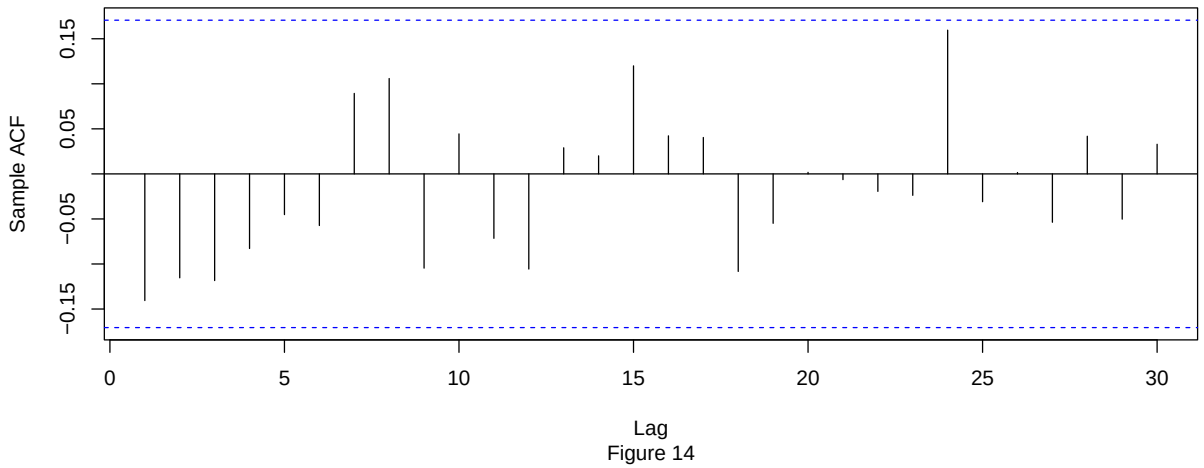


Figure 14

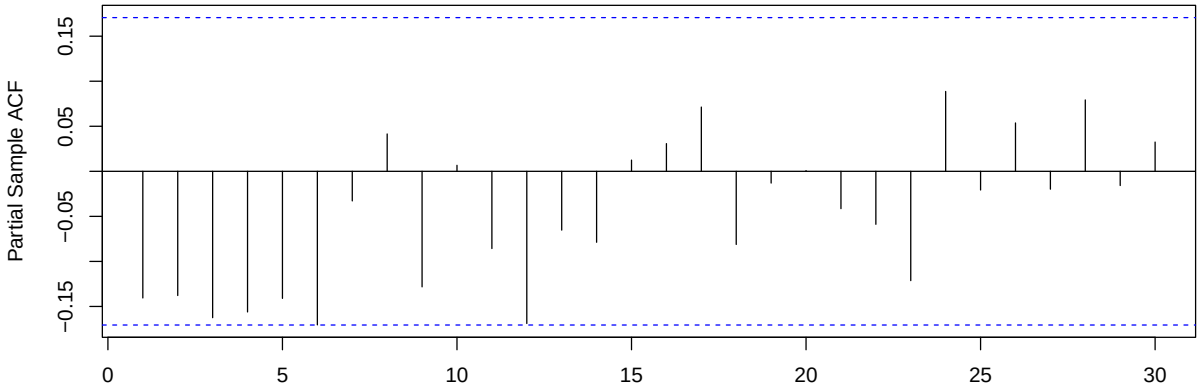


Figure 15

The ACF and PACF plot does not indicate alot.
However, we can find the optimal value of p and q by finding orders with the lowest AIC and BIC scores. After testing for this, p = 2 and q = 2. The above plots do not indicate any seasonality too.
Therefore, we choose these parameters as our first model's parameters.

Fitting and Diagnostics

Before deploying the model for real-world analysis, the model needs to achieve certain conditions on the training data itself, such as the difference between the training data predictions and actual training data must be a white noise process.
Let $\hat{y}$ be training data predictions, and y be actual training data.
The difference: $\hat{y} - y = w$
Where w is drawn from a white noise distribution, this is so that our predictions only deviate from actual observation due to some random effect and not from some other influence.

The difference between $\hat{y}$ and y is called residuals, and the residuals should have a normal distribution due to the powerful properties of Maximum Likelihood Estimation. However, achieving this result in a real-world dataset can be very challenging. As mentioned in the Introduction, certain global events can influence the VIX, which can be very hard to include in the model and predict.
Specific tests let us judge our model based on the criteria stated above, but in the end, what model needs to be used is our judgment, which must be logically decided for the particular problem.

Model 1: ARMA(2, 2)
In order to test if the residuals follow a white noise process, we can plot the ACF and PACF of residuals, the time plot of residuals, and the Ljung-Box test provides a statistical inference on the corrrelation of residuals.
If the time plot of residuals looks random without any trend, we can assume it is white noise.
If the ACF and PACF are within the bounds, we can deduce that the residuals are white noise.
If the Ljung-Box test p-value is more significant than 0.01, we infer that the residuals are uncorrelated.

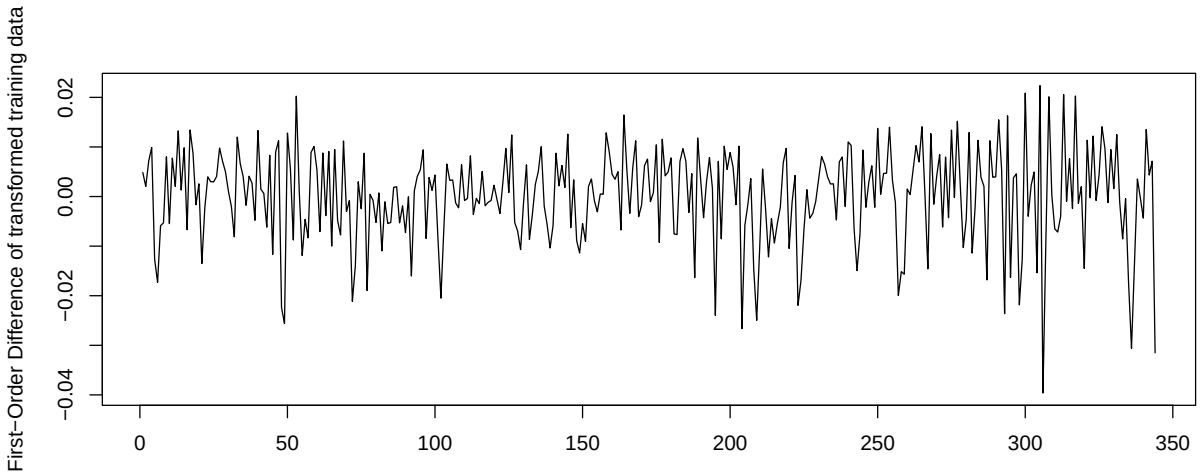


Figure 16

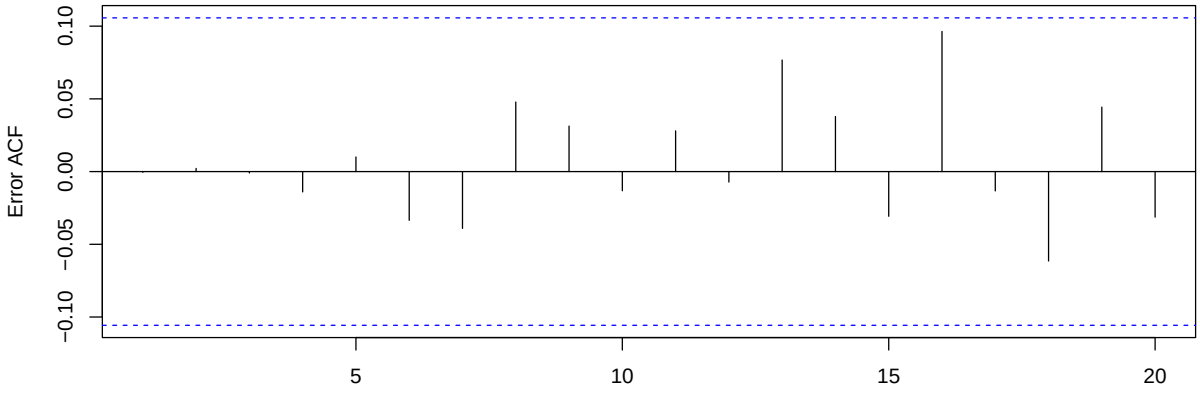


Figure 17

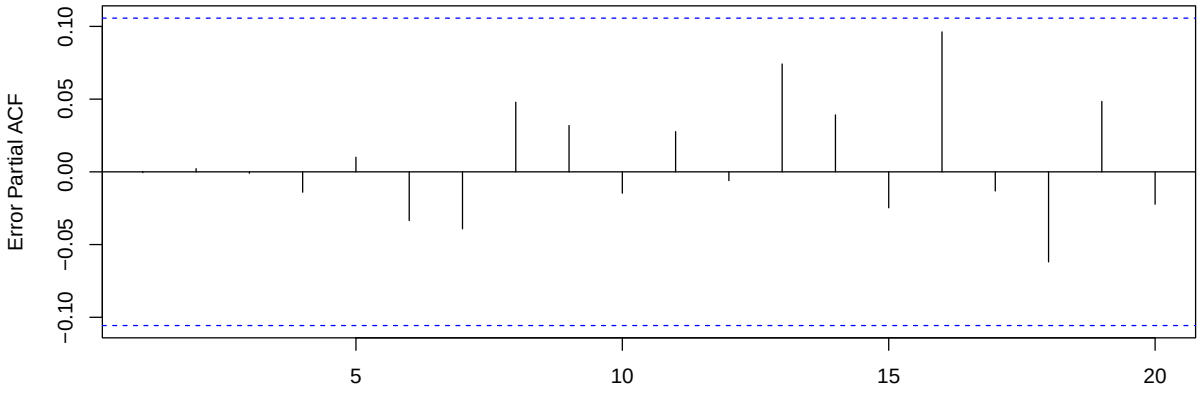


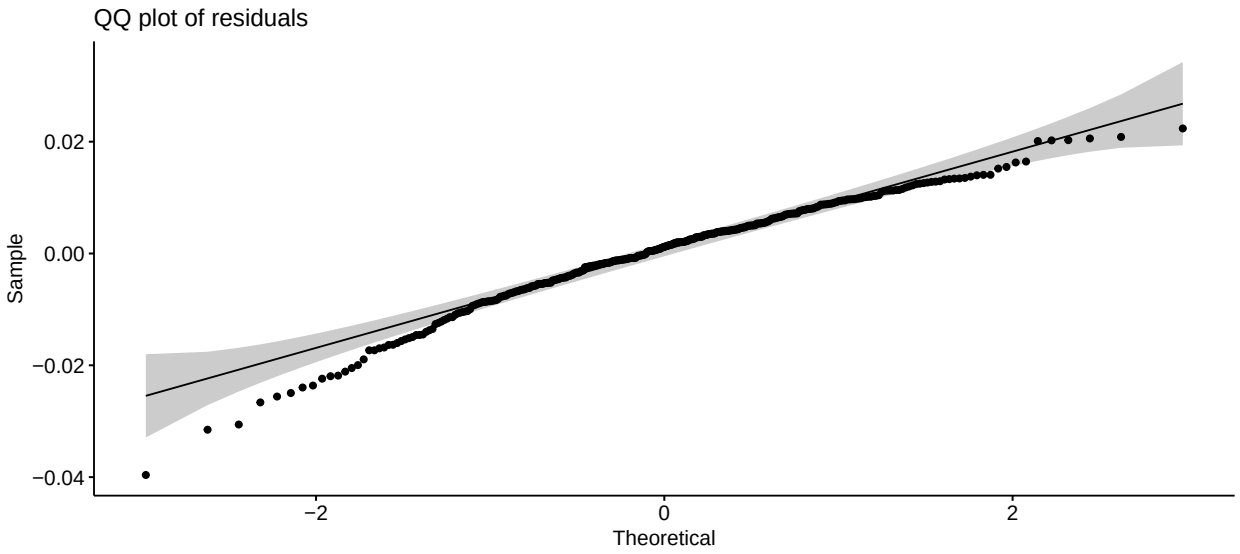
Figure 18

```
##
##  Box-Pierce test
##
## data:  residuals(y_3.fit1)
## X-squared = 9.6294e-05, df = 1, p-value = 0.9922
```

From the above, we can conclude that the residuals are white noise process and is uncorrelated.

In order to test if the white noise process is drawn from a normal distribution, we can plot the QQ plot, and the Shapiro-Wilk test provides a statistical inference of the residuals' distribution.
If the residuals lie within the line, we can deduce that residuals are normally distributed.
If the Shapiro-Wilk test p-value is more than 0.01, we infer that the residuals follow a white noise process.
But this test is very sensitive to large dataset and often results in a less than 0.01 p-value.

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(y_3.fit1)
## W = 0.971, p-value = 2.22e-06
```



From the above, we can conclude that the residuals is not normally distributed from shapiro-wilk test, but in the QQ plot of residuals roughly falls into the bounds.

Model 2: ARMA(2, 2)

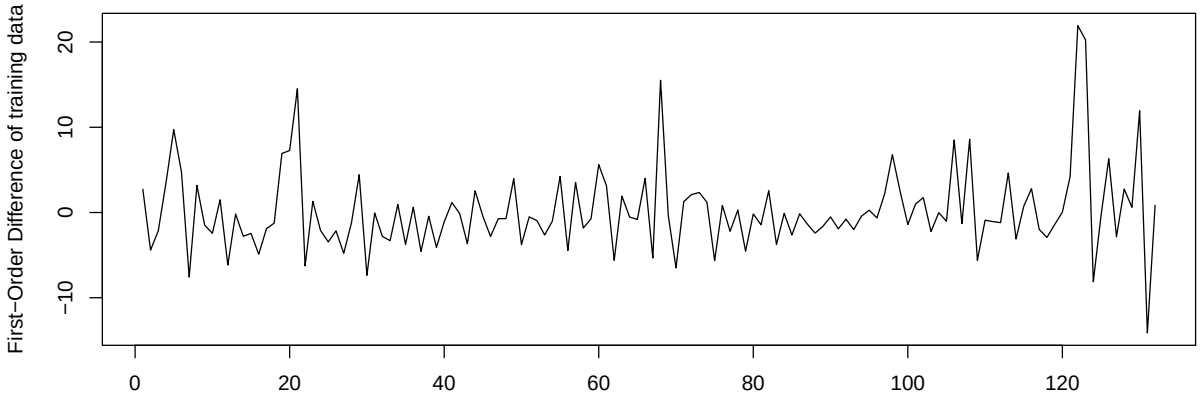


Figure 19

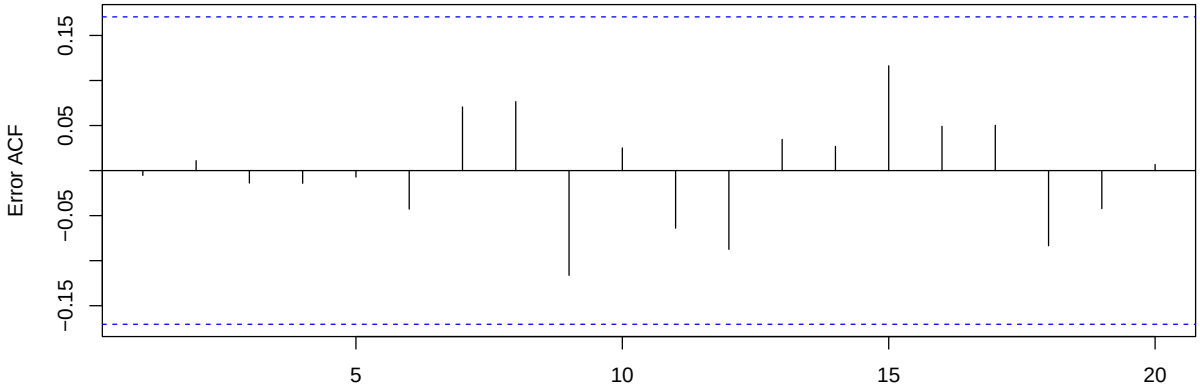


Figure 20

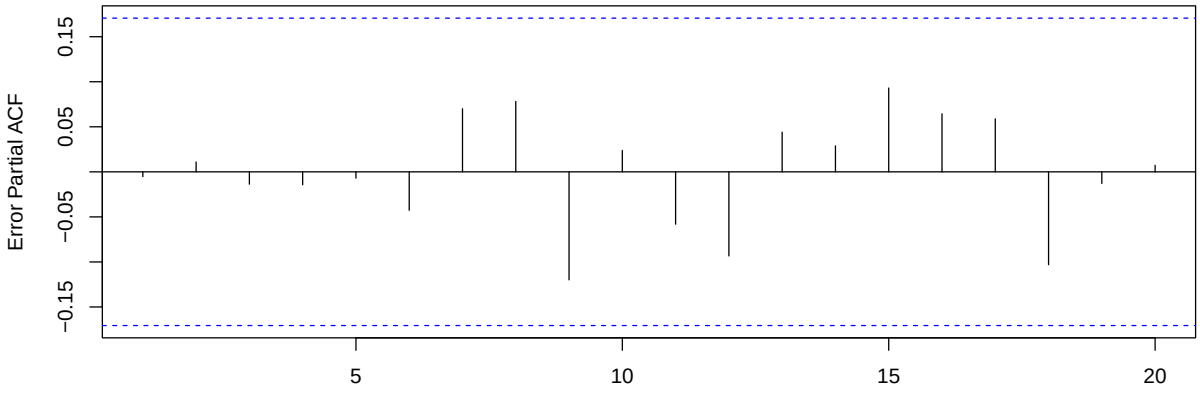
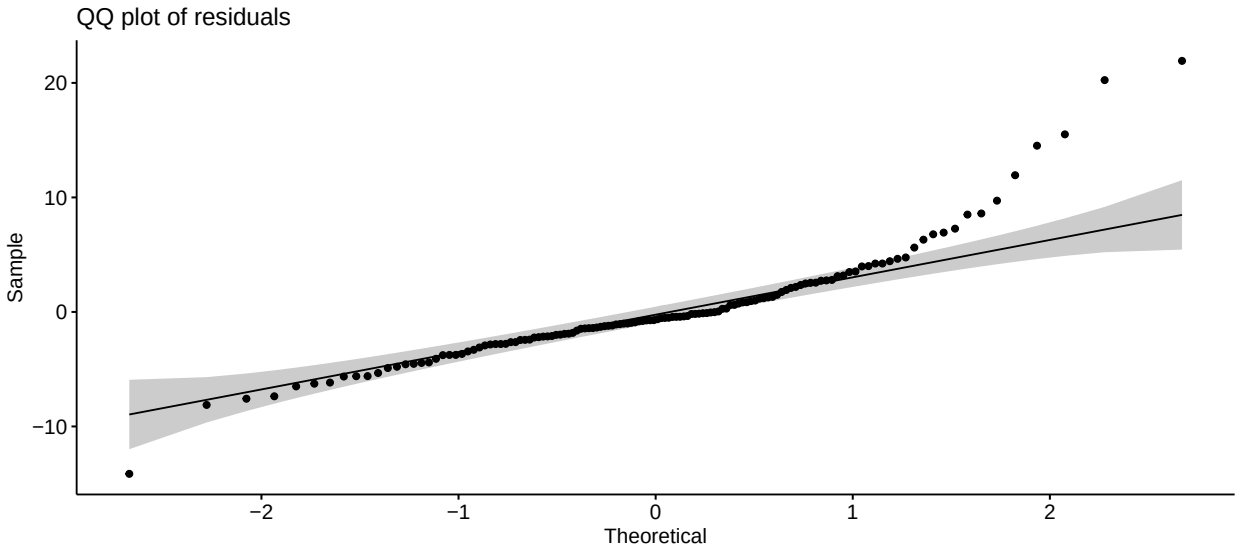


Figure 21

```
##
##  Box-Pierce test
##
## data:  residuals(y_4.fit1)
## X-squared = 0.0037978, df = 1, p-value = 0.9509
```

From the above we can conclude that the residuals are white noise process and it is uncorrelated.

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(y_4.fit1)
## W = 0.88009, p-value = 6.38e-09
```



From the above, the Shapiro-Wilk fails, and in the QQ-plot there is a slight deviation from normality at right.

From the above two models, both seem to deviate from normality, model 2 more than model 1. It is desirable to have normally distributed residuals, due to time constraints I was unable to figure a way out of this. Nevertheless, considering the context of this analysis and an overview of the dataset, it makes more sense to use model 2 as our final model for predictions makes more sense. As mentioned in the introduction, data from a different period may be useless for future predictions from the present.

Forecasting

One of the primary objectives of building a model for a time series is to be able to forecast the values for that series at future times. Of equal importance is the assessment of the precision of those forecasts [4]. We will test the goodness of the forecast by comparing it to the test dataset, which we had kept apart.

The following formula gives the forecast of the time series: $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + W_t + \theta_1 W_{t-1} + \theta_2 W_{t-2}$
Where ϕ_1 and ϕ_2 are coefficients of the AR part of the ARMA model, and θ_1 and θ_2 are the coefficients of the MA part of the ARMA model.

We will be forecasting 1 step; therefore, we will use every data point from 2010 January to the time we are in now and iterate through 2021, December 01, which forecasts the value of VIX at 2021, December 23. One way to estimate how correct our prediction is by calculating the Root Mean Square Error (RMSE),

RMSE $= \sqrt{\frac{\sum_{i=1}^{N}(\hat{x_t}-x_t)^2}{N}}$
where $\hat{x_t}$ is predicted value and x_t is the actual value.

We can develop a prediction interval that gives a bound for our prediction; we will be using a 95% confidence interval,
Prediction interval: $(\hat{x_{t+1|t}} - 1.96 * \sigma_1, \hat{x_{t+1|t}} + 1.96 * \sigma_1)$
where σ_1 is the standard error of prediction.

It is crucial to have an accurate prediction interval, and a low RMSE value as these predictions could decide customers' behavior in the stock exchange and have a real-life stake.

Taking a time plot of the forecasted value and actual values of the test data shows us that our predictions are following the predictions and nearly only miss some points.
Here the red dots indicate forecast and blue dots indicate actual values of VIX.

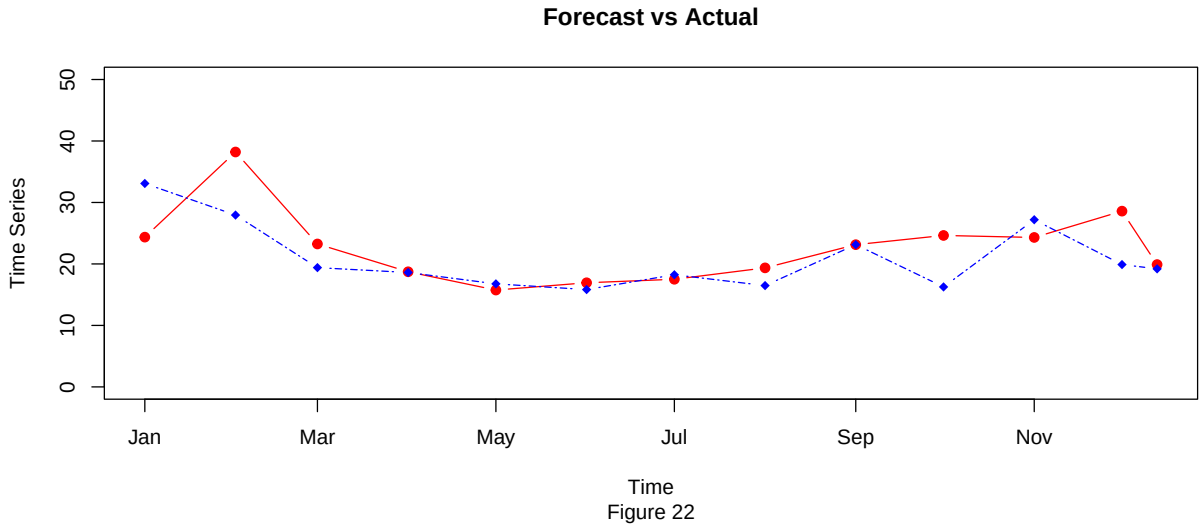


Figure 22

The RMSE value of the model = 7.622863.
The RMSE value is relatively high, whereas according to [5], the RMSE value the study got for ten days forecast was 5.2.

A time plot of the forecasted value and actual values of the test data with the prediction bounds shows that most of the values fall within the boundaries.
Here the black line indicate actual values of VIX, red line indicate the forecast values, and the blue shading is the prediction interval.

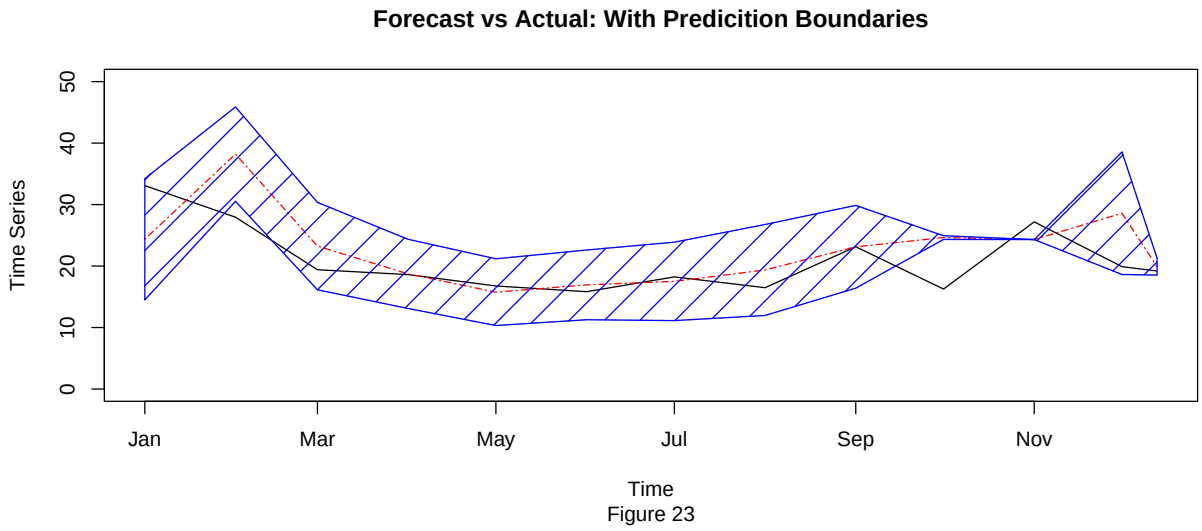


Figure 23

An overall time plot with the entire dataset with the forecast is as follows. Here black is the actual VIX values, red is the forecasted values, and the blue boundary is the predicition interval.

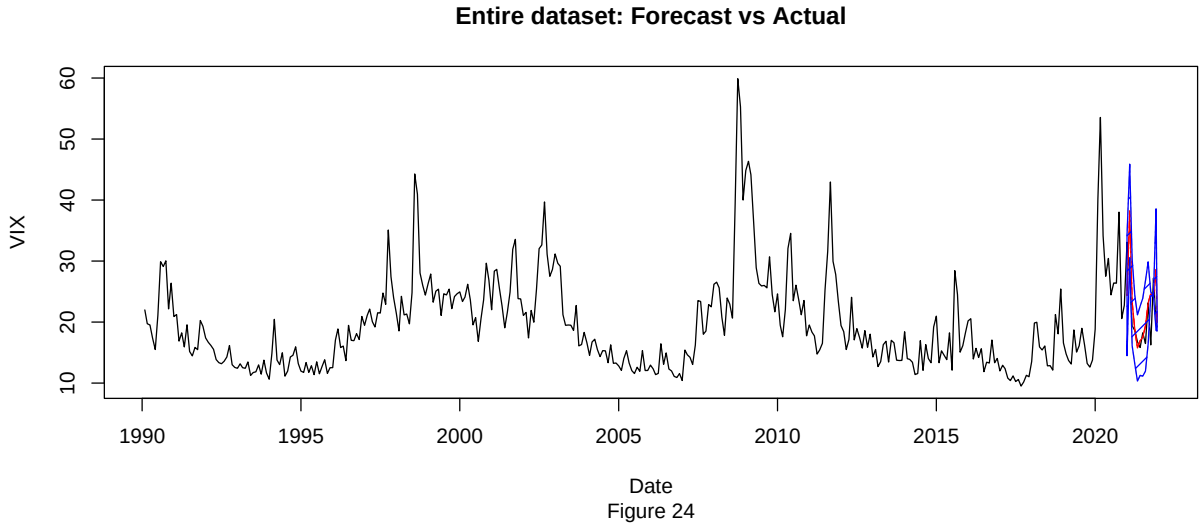


Figure 24

Discussion

From our forecast results, we can witness that our prediction of VIX using Time Series Analysis was successful to some degree. However, more than the forecast, it is the prediction interval that must be used by someone who is deciding on the implied volatility of the market as this provides a range of future value of reasonable accuracy.

However, our predictions are not picture-perfect from Figure 23; during the times 2021 September to November, our forecast deviates from the actual values. This deviation could indicate either that our model was inadequate for predicting or that there was some external geopolitical situation that caused a change from its usual trend. We could hypothesize if this were due to the Delta variant of Covid-19 that hit in 2021, August-November, but that is out of scope for our project.

One of the main issues faced in this project was getting the residuals to have a white noise process; this could

indicate the shortcomings of the proposed model for predicting VIX.

Bibliography

[1] https://www.wallstreetmojo.com/realized-volatility/
[2] https://www.investopedia.com/terms/i/iv.asp/
[3] https://www.investopedia.com/terms/v/vix.asp/
[4] Time Series Analysis With Applications in R. Jonathan D. Cryer, Kung-Sik Chan.
[5] https://www.etf.com/publications/journalofindexes/joi-articles/10096-realized-volatility-indexes.html/page/0/3/

Appendices

The R codes used for this project are mentioned below, I followed Box-Jenkins workflow for analysis.

Import Libraries

```
library(ggplot2)
library(splitTools)
library(tseries)
library(aTSA)
library(TSA)
library(gdata)
library(psych)
library(dplyr)
library(gdata)
```

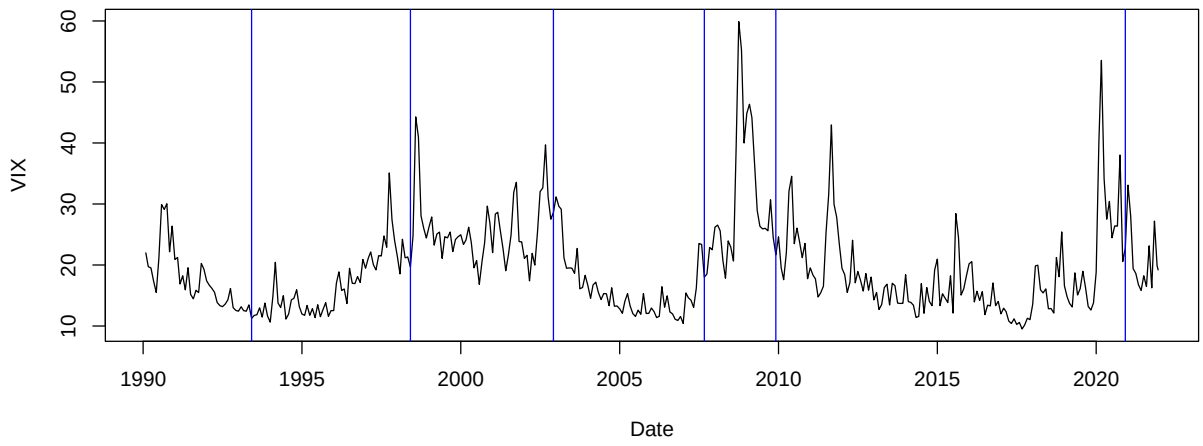
Import Dataset

```
setwd("C:/Users/nithi/OneDrive - University of Toronto/Year -4/STA457/project")
data = read.csv("Data.csv")
```

Data Preprocessing

```
data$ï..Date = as.Date(data$ï..Date)
names(data)[names(data) == "ï..Date"] <- "Date"
```

```
plot(data, type="l", col="Black", xlim=c(as.Date("1990-02-01"), as.Date("2021-12-13")))
abline(v=as.Date("1993-06-01"), col="blue")
abline(v=as.Date("1998-06-01"), col="blue")
abline(v=as.Date("2002-12-01"), col="blue")
abline(v=as.Date("2007-09-01"), col="blue")
abline(v=as.Date("2009-12-01"), col="blue")
abline(v=as.Date("2020-12-01"), col="blue")
```

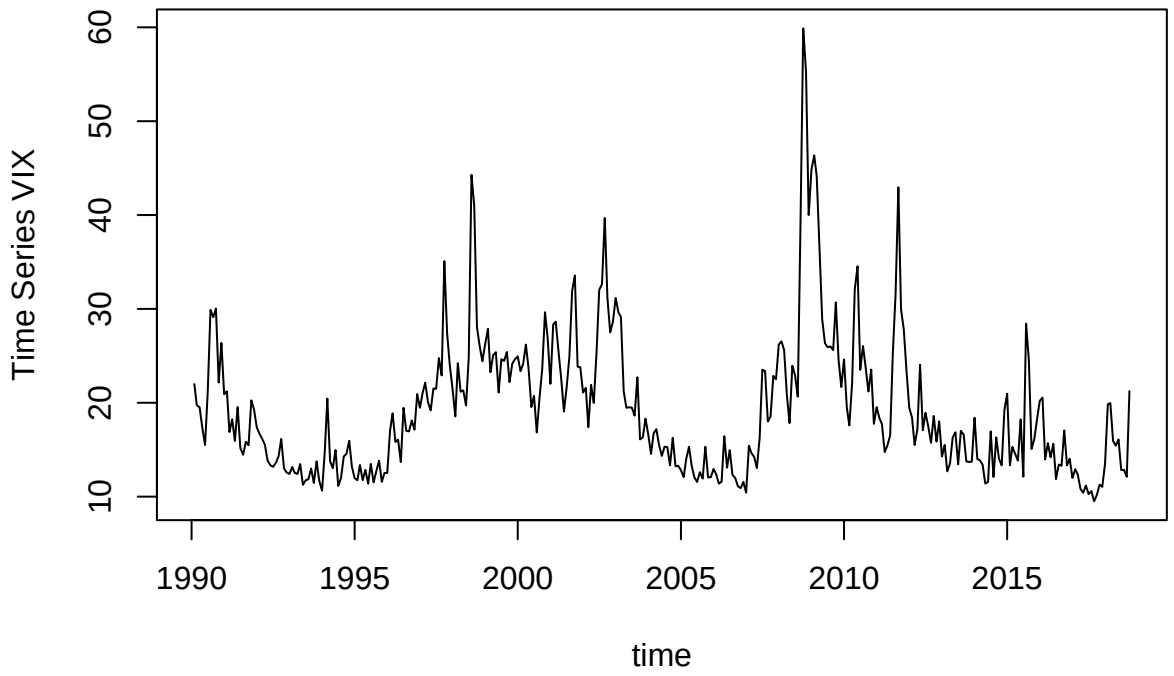



```
seg_start = which(data$Date==as.Date("2009-12-01"))
data_seg = subset(data, Date>=as.Date("2009-12-01"))
train_seg = subset(data_seg, Date<=as.Date("2020-12-01"))
test_seg = subset(data_seg, Date>as.Date("2020-12-01"))
```

```
idx = 1:as.integer(dim(data)[1] * 0.9)
train = data[idx, ]
test = data[-idx, ]
```

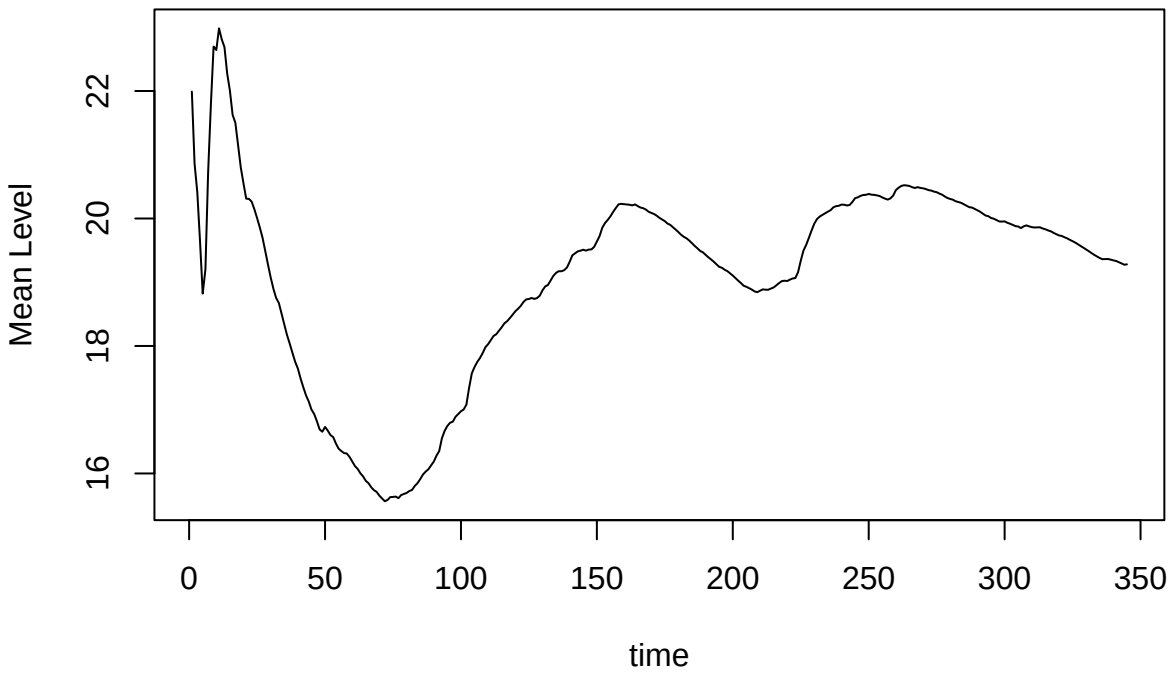
Model Specification

```
cummeanx = cumsum(train$VIX) / seq_along(train$VIX)
plot(train$Date, train$VIX, xlab="time", ylab="Time Series VIX", type="l")
```

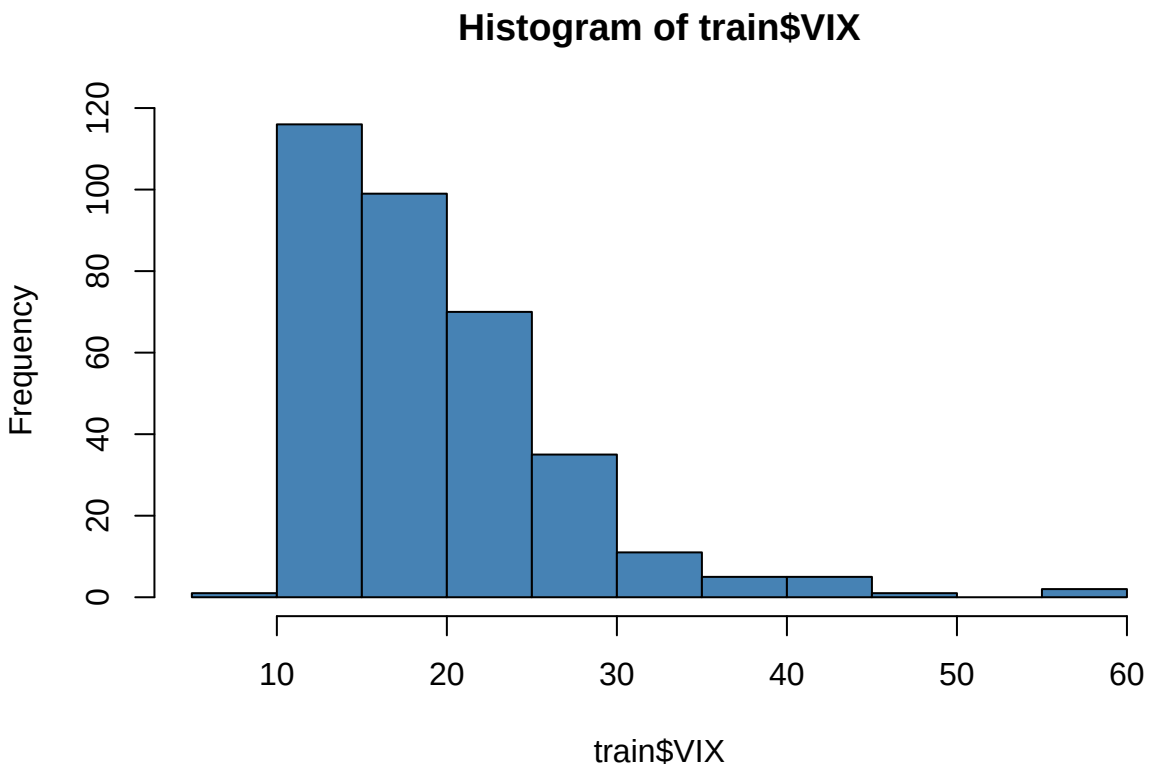


No segmentation

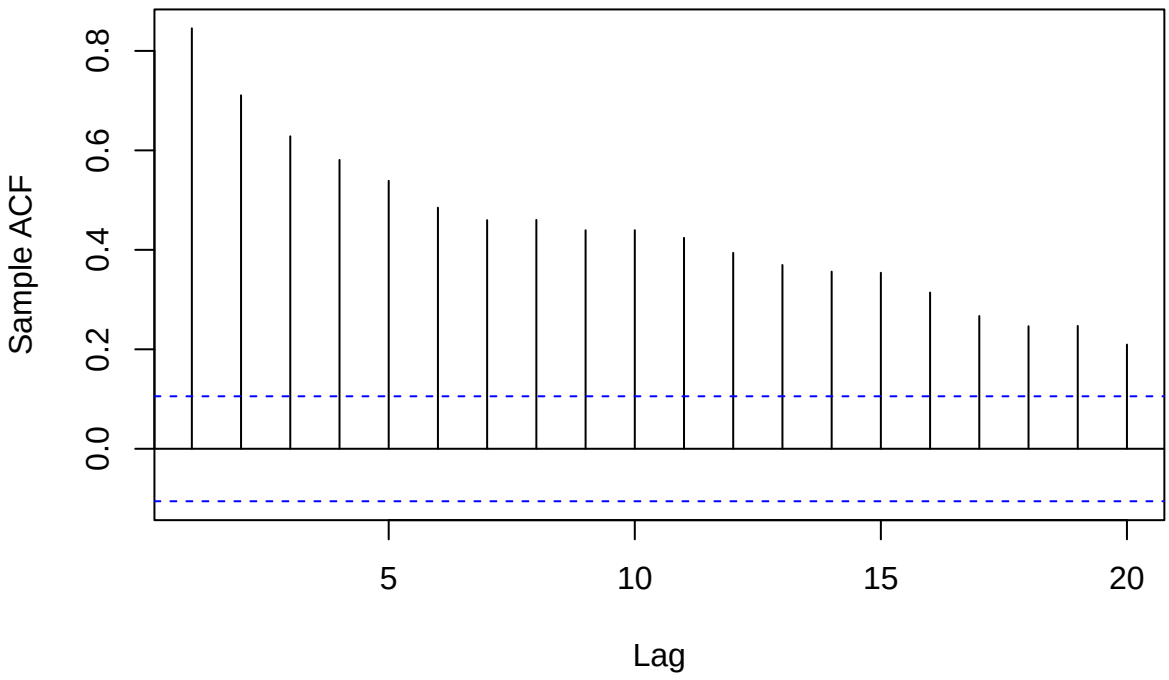
```
plot(cummeanx, xlab="time", ylab="Mean Level", type="l")
```



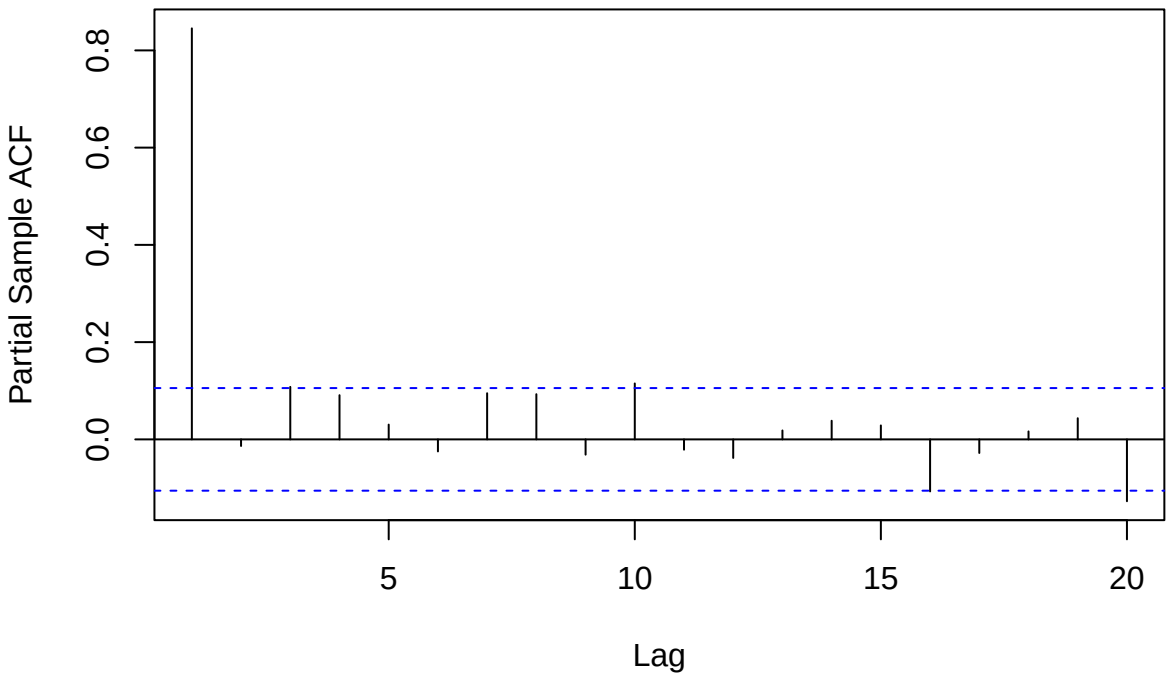

```
hist(train$VIX, col="Steelblue")
```




```
acf(train$VIX, lag.max = 20, na.action = na.pass, xlab="Lag", ylab="Sample ACF", main="")
```




```
pacf(train$VIX, lag.max = 20, na.action = na.pass, xlab="Lag", ylab="Partial Sample ACF", main="")
```




```
VIX.adf = adf.test(train$VIX)
```

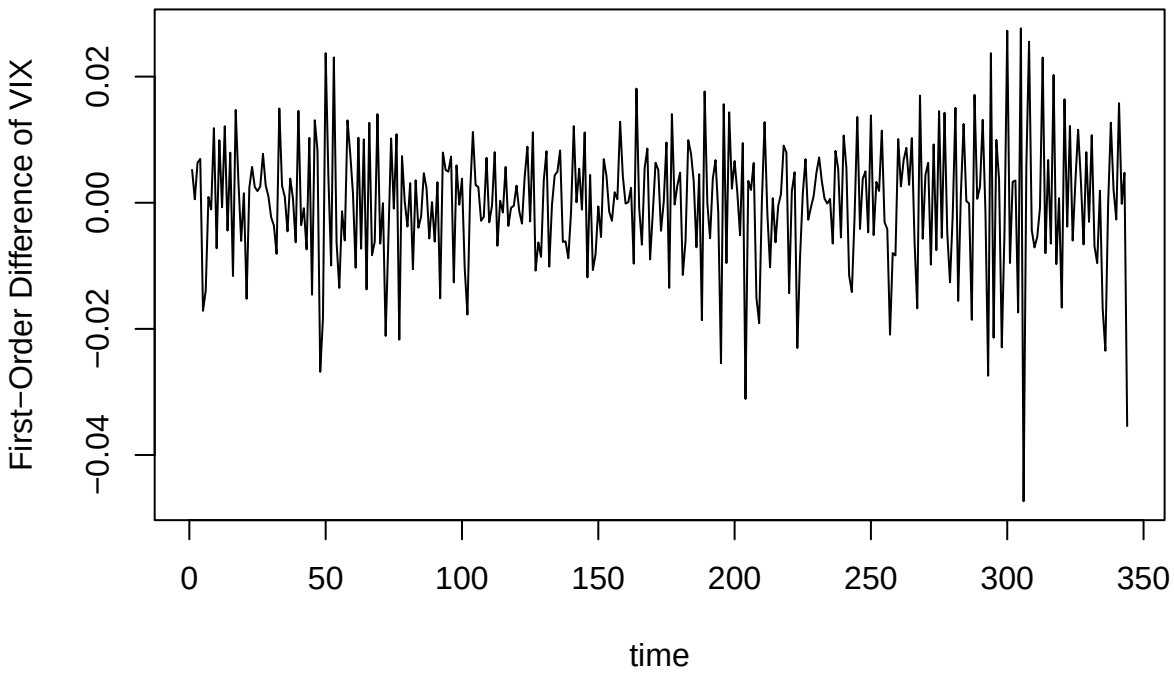
```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -1.88  0.0611
## [2,]   1 -1.75  0.0808
## [3,]   2 -1.48  0.1502
## [4,]   3 -1.27  0.2250
## [5,]   4 -1.16  0.2656
## [6,]   5 -1.20  0.2505
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -5.35    0.01
## [2,]   1 -5.21    0.01
## [3,]   2 -4.48    0.01
## [4,]   3 -3.96    0.01
## [5,]   4 -3.74    0.01
## [6,]   5 -3.74    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -5.35   0.010
## [2,]   1 -5.20   0.010
## [3,]   2 -4.47   0.010
```

```
## [4,]   3 -3.96   0.011
## [5,]   4 -3.74   0.022
## [6,]   5 -3.74   0.022
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
VIX.kpss = kpss.test(train$VIX)
```

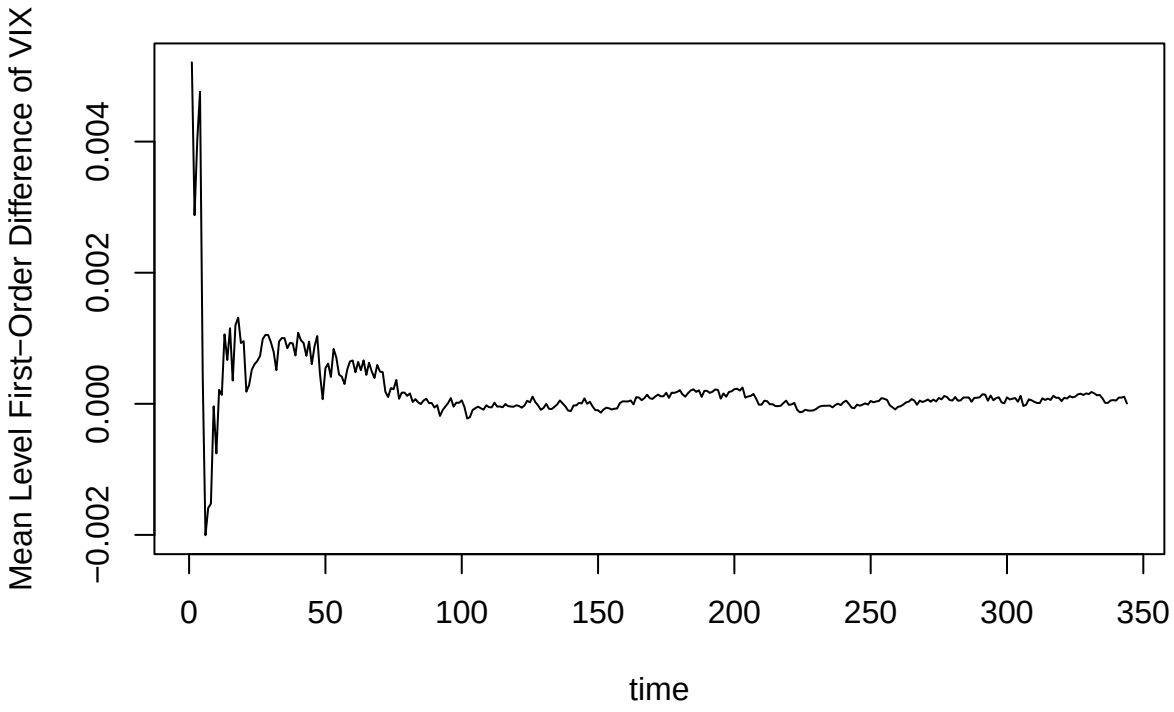
```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag stat p.value
##    4 1.58  0.0581
## -----
##  Type 2: with drift no trend
##  lag  stat p.value
##    4 0.153     0.1
## -----
##  Type 1: with drift and trend
##  lag  stat p.value
##    4 0.152  0.0447
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

```
y_3 = diff((1/train$VIX), differences=1)

cummeany_3 = cumsum(y_3) / seq_along(y_3)

plot(y_3, xlab="time", ylab="First-Order Difference of VIX", type="l")
```

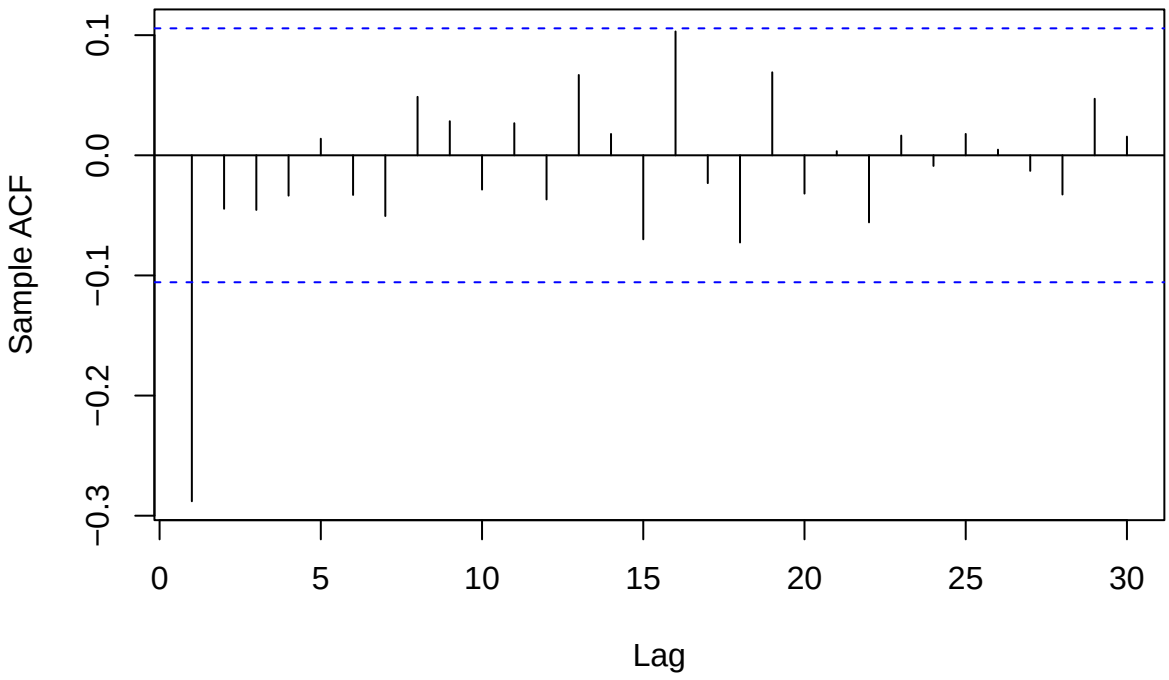


Model 1

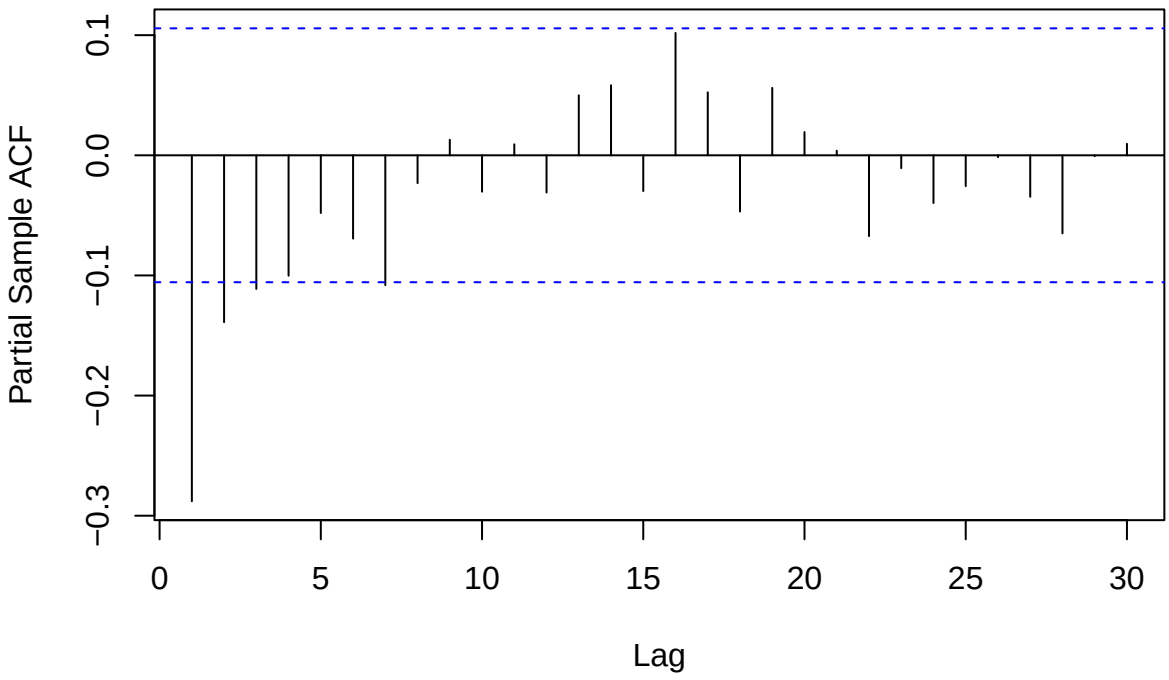
```
plot(cummeany_3, xlab="time", ylab="Mean Level First-Order Difference of VIX", type="l")
```




```
acf(y_3, lag.max = 30, na.action = na.pass, xlab="Lag", ylab="Sample ACF", main="")
```




```
pacf(y_3, lag.max = 30, na.action = na.pass, xlab="Lag", ylab="Partial Sample ACF", main="")
```




```
y_3.adf = adf.test(y_3)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -24.68    0.01
## [2,]   1 -16.94    0.01
## [3,]   2 -13.94    0.01
## [4,]   3 -12.31    0.01
## [5,]   4 -10.63    0.01
## [6,]   5  -9.82    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -24.65    0.01
## [2,]   1 -16.91    0.01
## [3,]   2 -13.92    0.01
## [4,]   3 -12.29    0.01
## [5,]   4 -10.61    0.01
## [6,]   5  -9.81    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -24.61    0.01
## [2,]   1 -16.88    0.01
## [3,]   2 -13.89    0.01
```

```
## [4,]   3 -12.27    0.01
## [5,]   4 -10.60    0.01
## [6,]   5  -9.79    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
y_3.kpss = kpss.test(y_3)
```

```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag   stat p.value
##    4 0.0306     0.1
## -----
##  Type 2: with drift no trend
##  lag   stat p.value
##    4 0.0271     0.1
## -----
##  Type 1: with drift and trend
##  lag   stat p.value
##    4 0.0276     0.1
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

```
y_3.eacf = eacf(y_3, ar.max=5, ma.max=5)
```

```
## AR/MA
##   0 1 2 3 4 5
## 0 x o o o o o
## 1 x o o o o o
## 2 x o o o o o
## 3 x o x o o o
## 4 x x x o o o
## 5 x o x o o o
```

```
y_3.aic = matrix(0, 5, 5)
y_3.bic = matrix(0, 5, 5)

for (i in 0:4) for (j in 0:4){
  y_3.fit = arima(y_3, order=c(i, 0, j), method="ML", include.mean=TRUE)
  y_3.aic[i+1, j+1] = AIC(y_3.fit)
  y_3.bic[i+1, j+1] = BIC(y_3.fit)
}

y_3.aic_vec = sort(unmatrix((y_3.aic), byrow=FALSE))[1:13]
y_3.bic_vec = sort(unmatrix((y_3.bic), byrow=FALSE))[1:13]
```

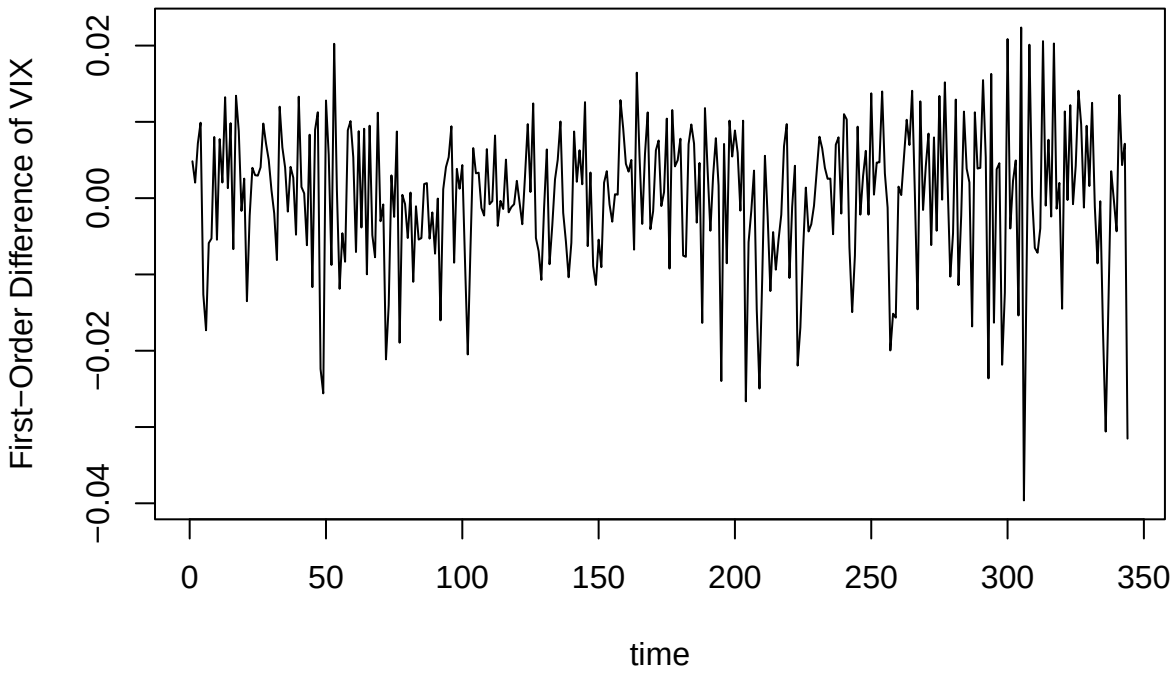
```
y_3.aic_vec
```

```
##     r2:c2     r3:c2     r2:c3     r4:c4     r1:c4     r4:c5     r1:c3     r2:c4
## -2216.559 -2215.276 -2215.169 -2214.976 -2214.908 -2214.195 -2214.117 -2213.417
##     r1:c5     r4:c2     r3:c3     r5:c4     r5:c2
## -2213.411 -2213.396 -2213.343 -2213.202 -2211.499
```

```
y_3.bic_vec
```

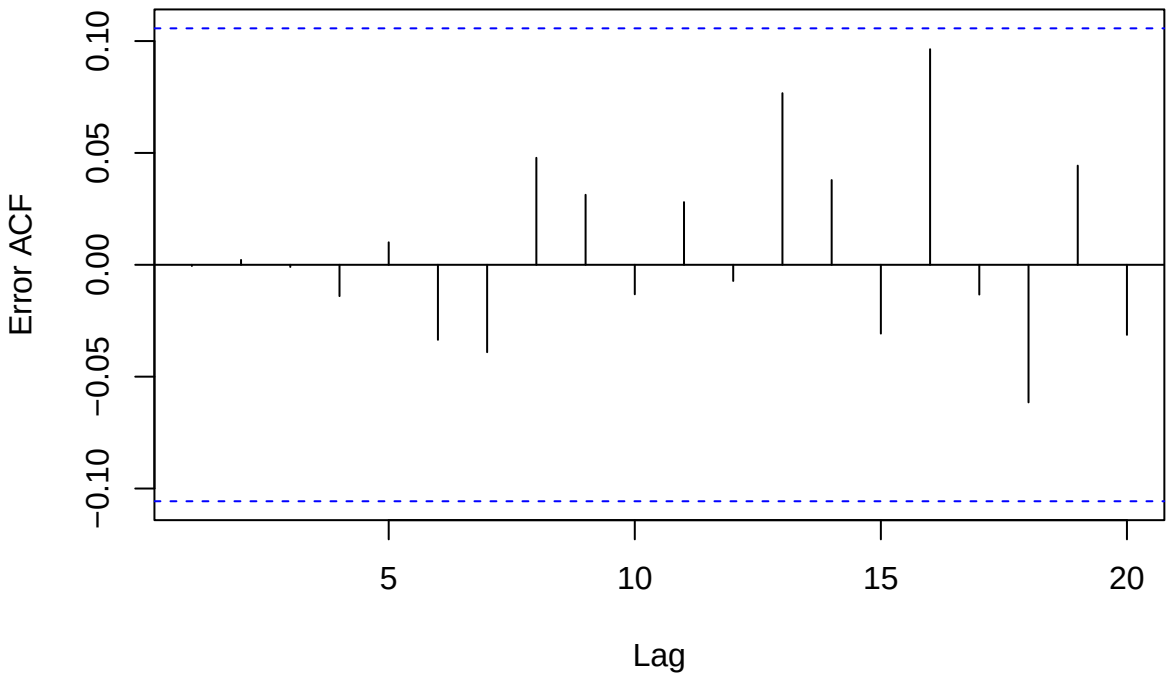
```
##     r2:c2     r1:c2     r1:c3     r3:c2     r2:c3     r1:c4     r2:c4     r1:c5
## -2201.197 -2199.736 -2198.754 -2196.073 -2195.965 -2195.704 -2190.373 -2190.367
##     r4:c2     r3:c3     r3:c1     r4:c1     r2:c1
## -2190.352 -2190.299 -2189.706 -2188.559 -2188.387
```

```
y_3.fit1 = arima(y_3, order=c(2, 0, 2), method="ML", include.mean = FALSE)
plot(residuals(y_3.fit1), xlab="time", ylab="First-Order Difference of VIX", type="l")
```

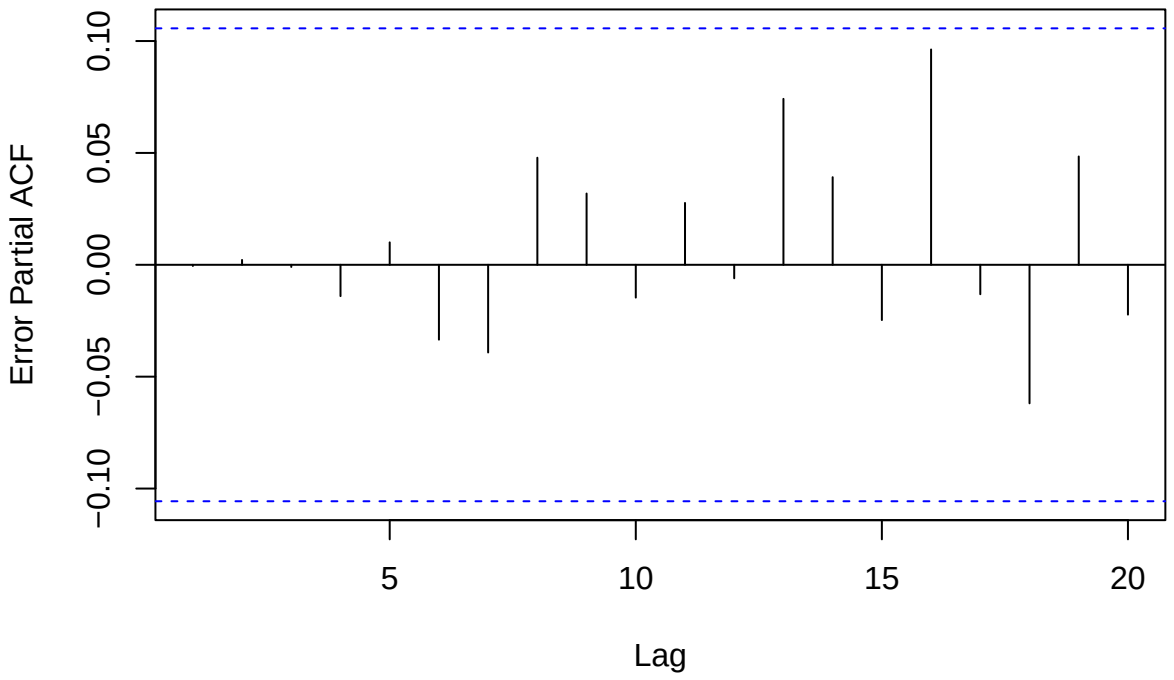


Model Diagnostics

```
acf(residuals(y_3.fit1), lag.max = 20, na.action = na.pass, xlab="Lag", ylab="Error ACF", main="")
```



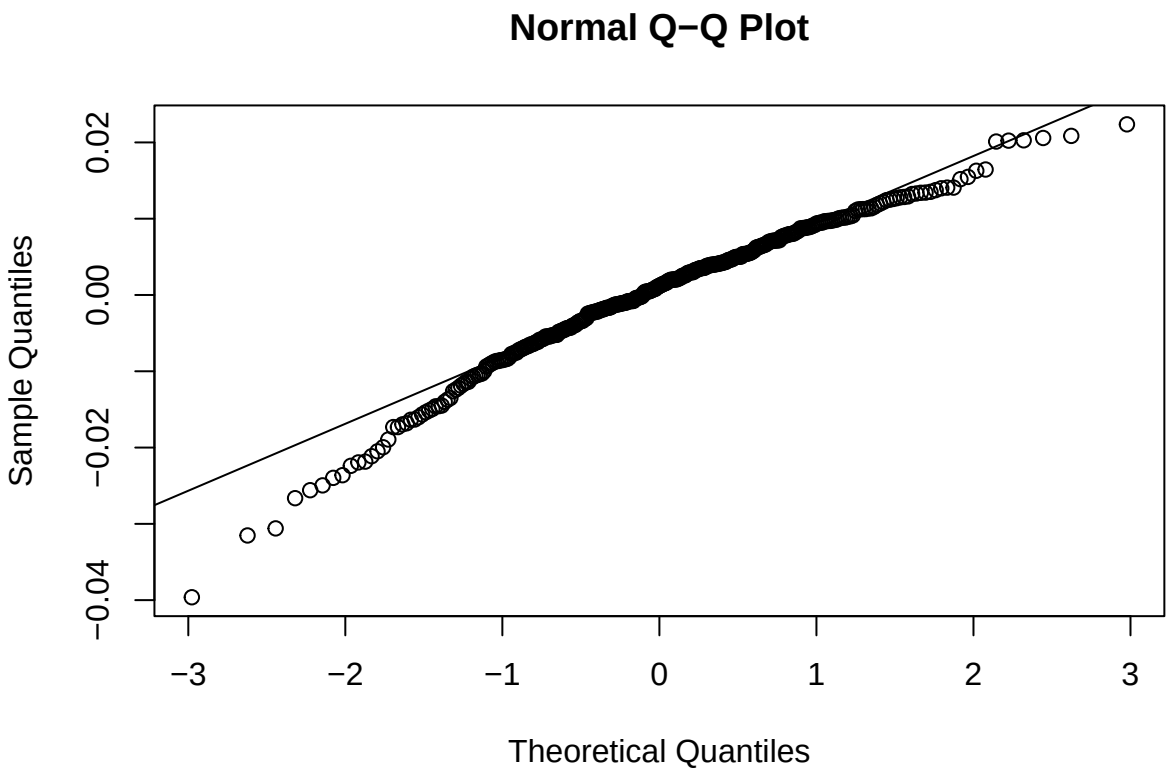

```
pacf(residuals(y_3.fit1), lag.max = 20, na.action = na.pass, xlab="Lag", ylab="Error Partial ACF", main=
```



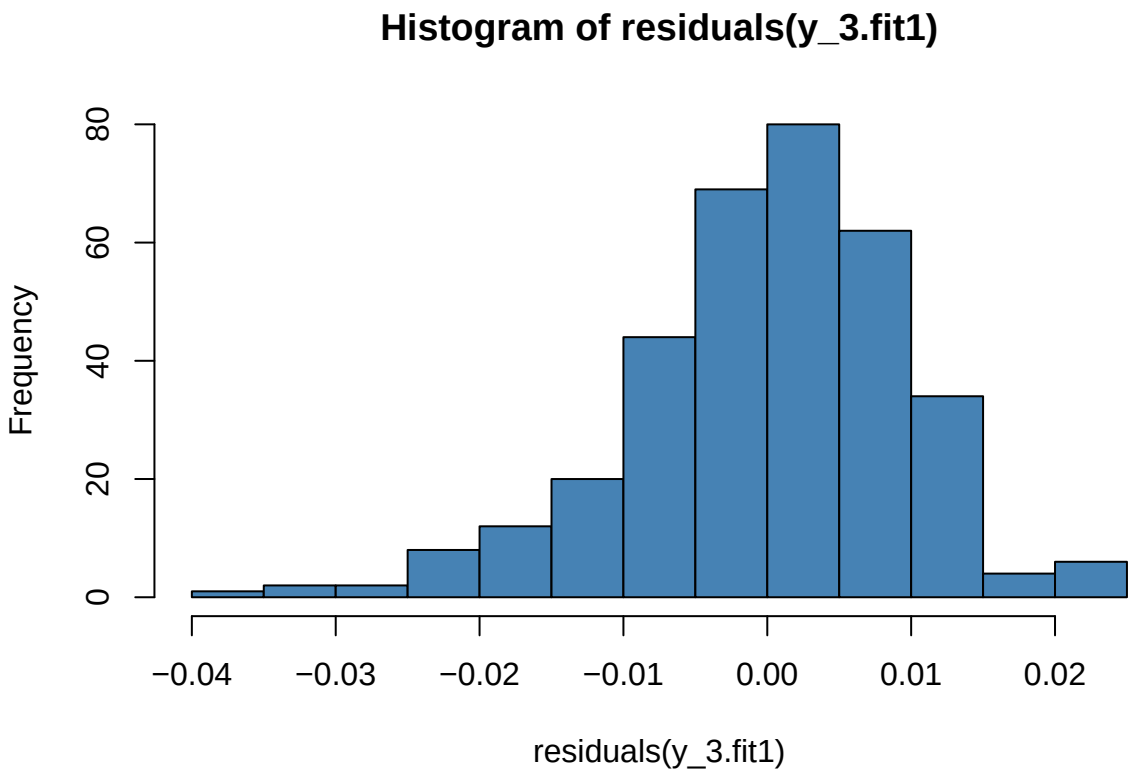

```
Box.test(residuals(y_3.fit1), lag=1, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  residuals(y_3.fit1)
## X-squared = 9.7136e-05, df = 1, p-value = 0.9921
```

```
qqnorm(residuals(y_3.fit1))
qqline(residuals(y_3.fit1))
```




```
hist(residuals(y_3.fit1), col='steelblue')
```



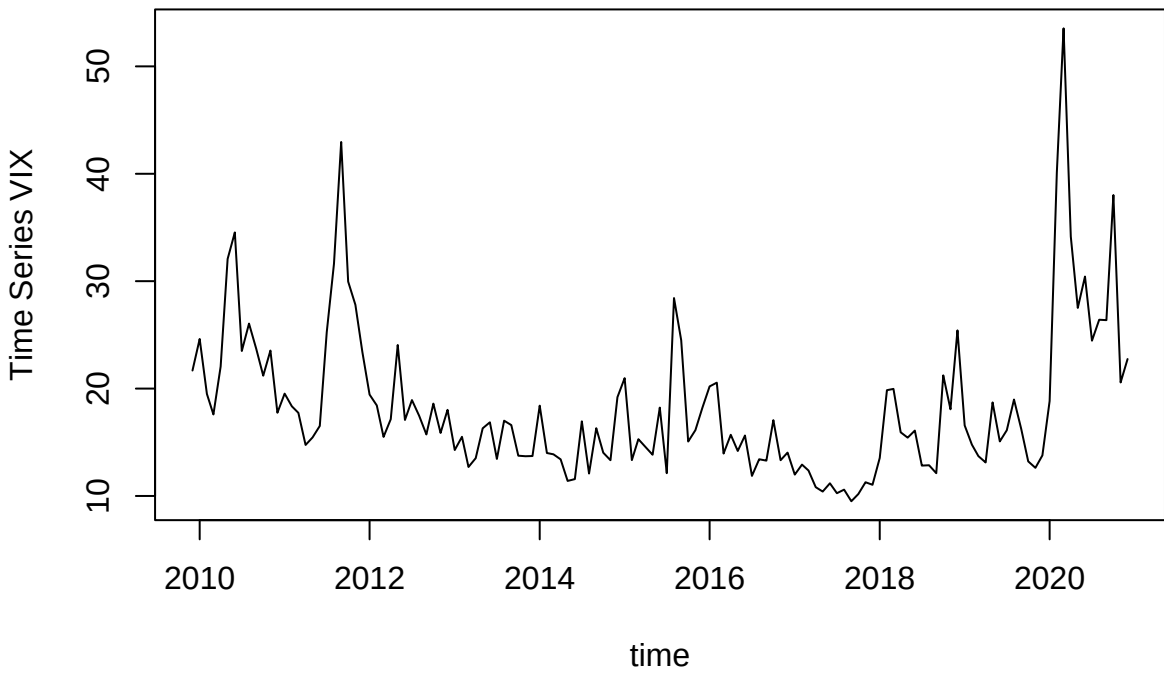

```
shapiro.test(residuals(y_3.fit1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(y_3.fit1)
## W = 0.971, p-value = 2.22e-06
```

```
ks.test(residuals(y_3.fit1), "pnorm")
```

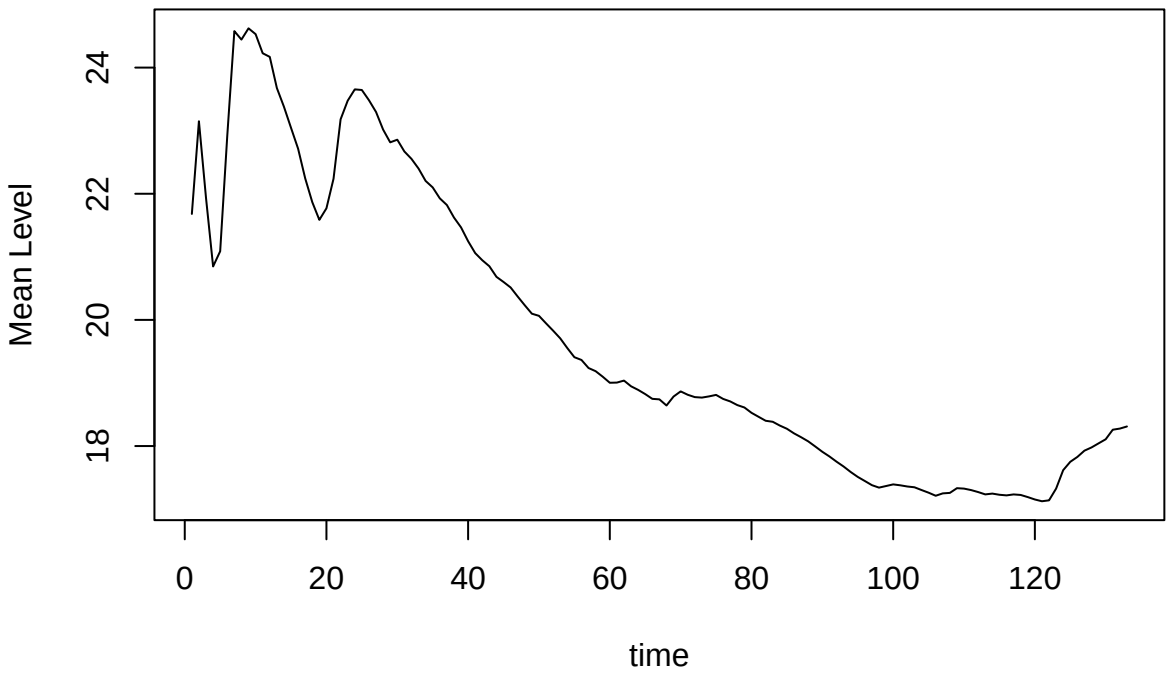
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  residuals(y_3.fit1)
## D = 0.49108, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
cummeanx = cumsum(train_seg$VIX) / seq_along(train_seg$VIX)
plot(train_seg$Date, train_seg$VIX, xlab="time", ylab="Time Series VIX", type="l")
```



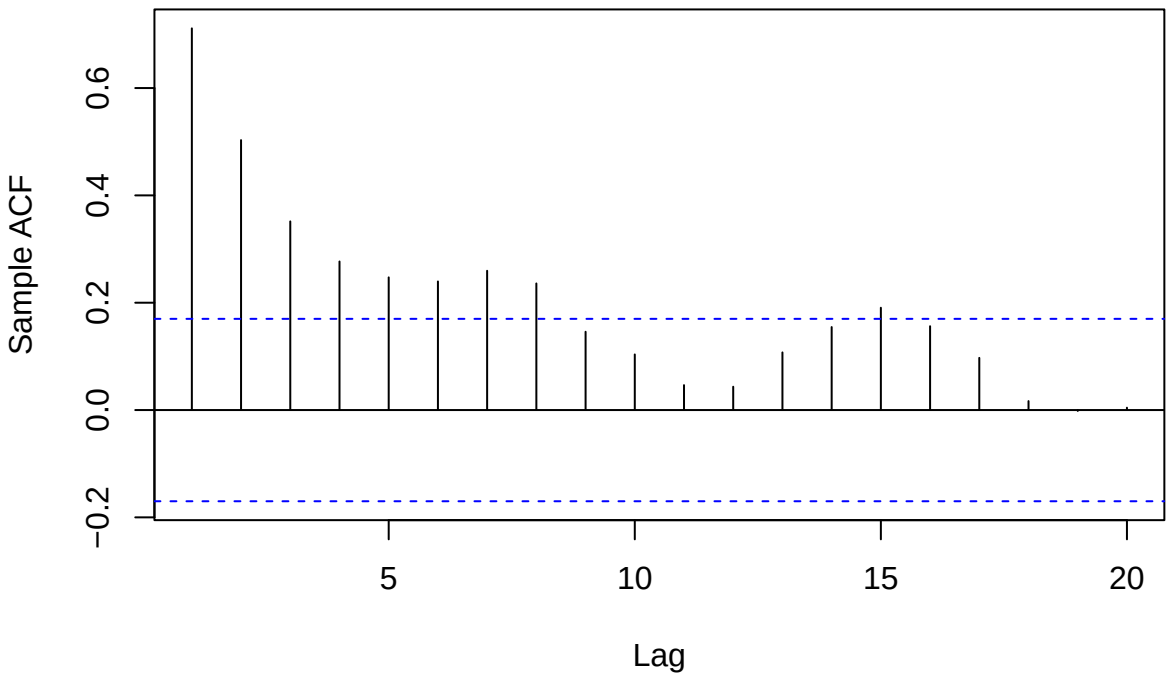
With segmentation

```
plot(cummeanx, xlab="time", ylab="Mean Level", type="l")
```

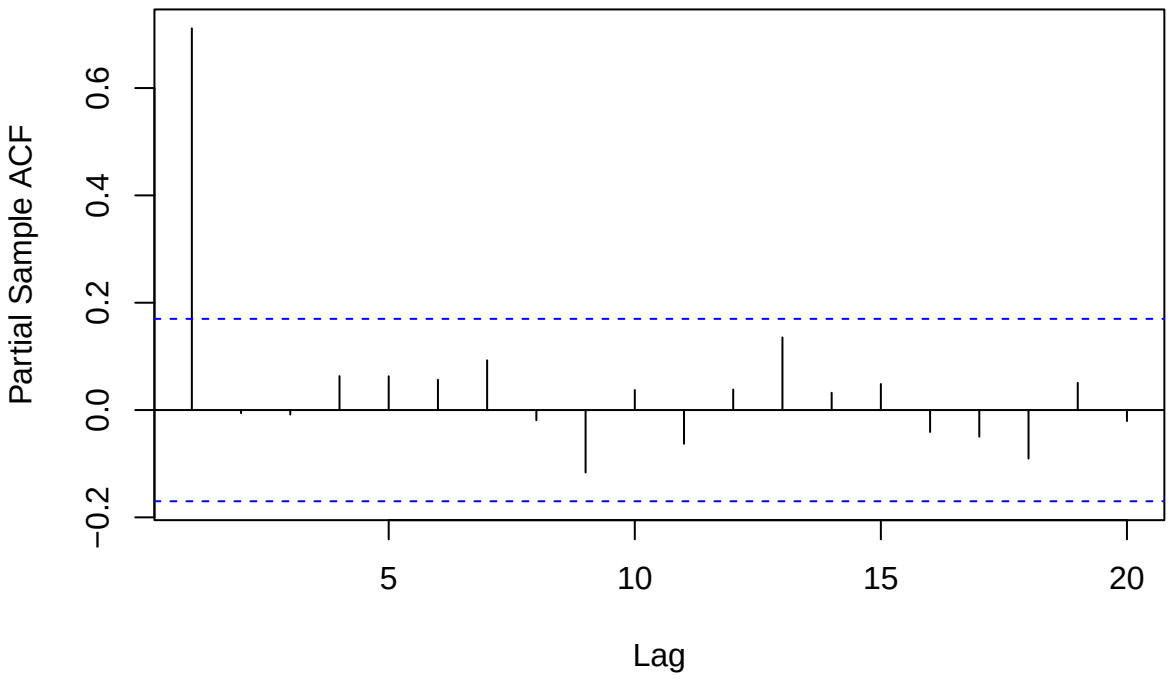



```
hist(train_seg$VIX, col="Steelblue")
```


```
acf(train_seg$VIX, lag.max = 20, na.action = na.pass, xlab="Lag", ylab="Sample ACF", main="")
```




```
pacf(train_seg$VIX, lag.max = 20, na.action = na.pass, xlab="Lag", ylab="Partial Sample ACF", main="")
```




```
VIX_seg.adf = adf.test(train_seg$VIX)
```

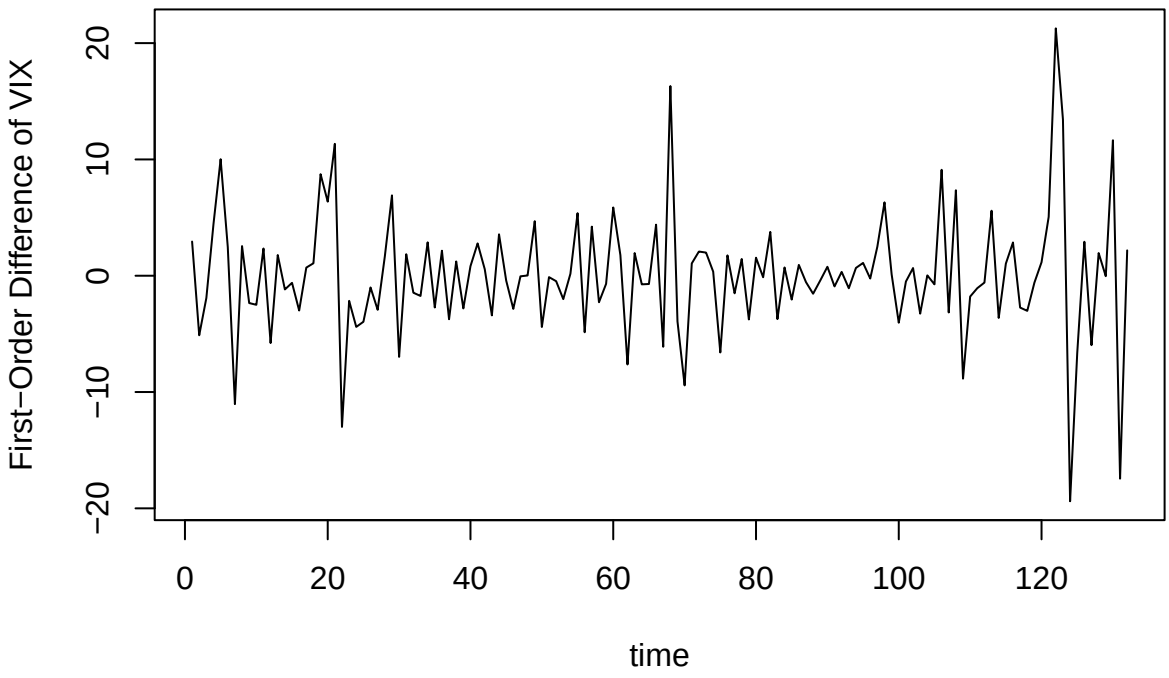
```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -1.548   0.123
## [2,]   1 -1.406   0.174
## [3,]   2 -1.089   0.288
## [4,]   3 -0.843   0.377
## [5,]   4 -0.738   0.414
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -4.66  0.0100
## [2,]   1 -4.33  0.0100
## [3,]   2 -3.89  0.0100
## [4,]   3 -3.39  0.0145
## [5,]   4 -2.99  0.0409
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -4.63  0.0100
## [2,]   1 -4.30  0.0100
## [3,]   2 -3.86  0.0182
## [4,]   3 -3.36  0.0636
## [5,]   4 -2.94  0.1854
```

```
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
VIX_seg.kpss = kpss.test(train_seg$VIX)
```

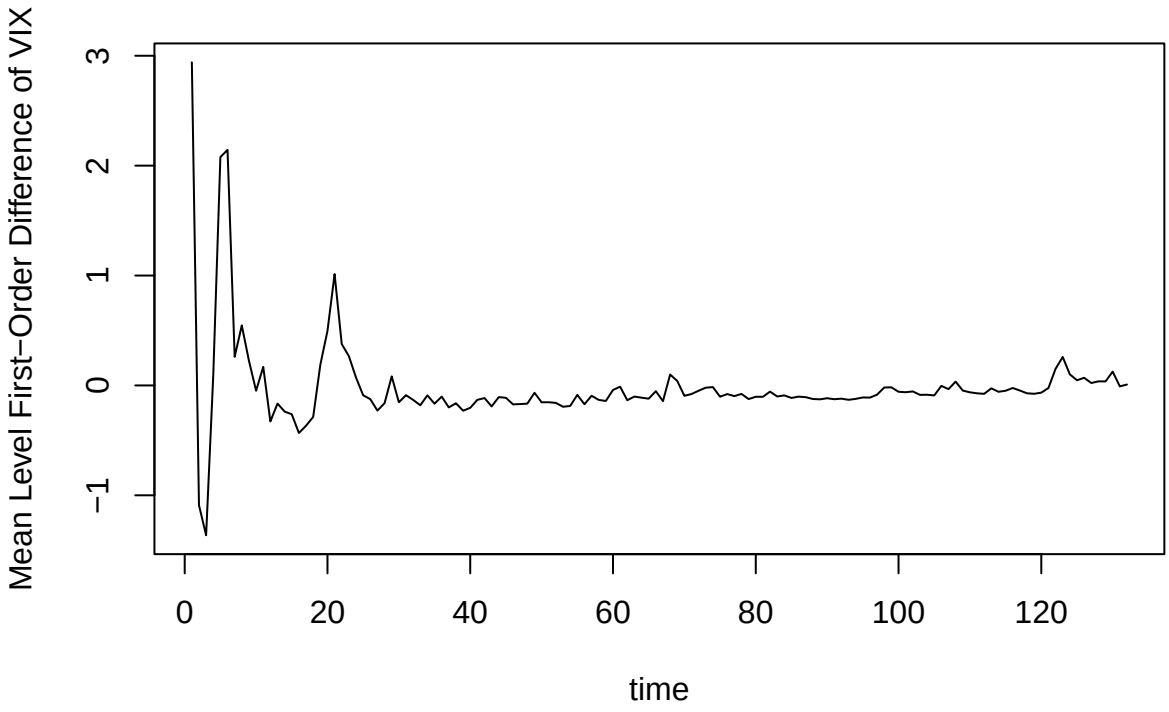
```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag stat p.value
##    2  0.8     0.1
## -----
##  Type 2: with drift no trend
##  lag  stat p.value
##    2 0.193     0.1
## -----
##  Type 1: with drift and trend
##  lag  stat p.value
##    2 0.192   0.019
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

```
y_4 = diff((train_seg$VIX), difference=1)

cummeany_4 = cumsum(y_4) / seq_along(y_4)

plot(y_4, xlab="time", ylab="First-Order Difference of VIX", type="l")
```

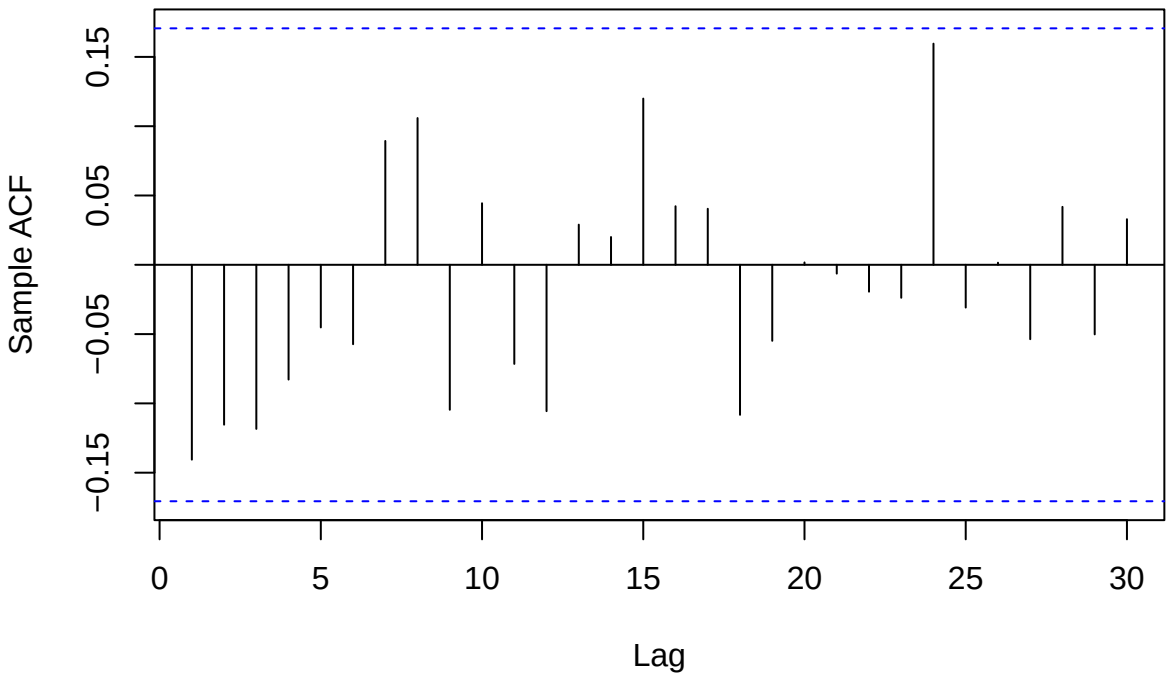


Model 2

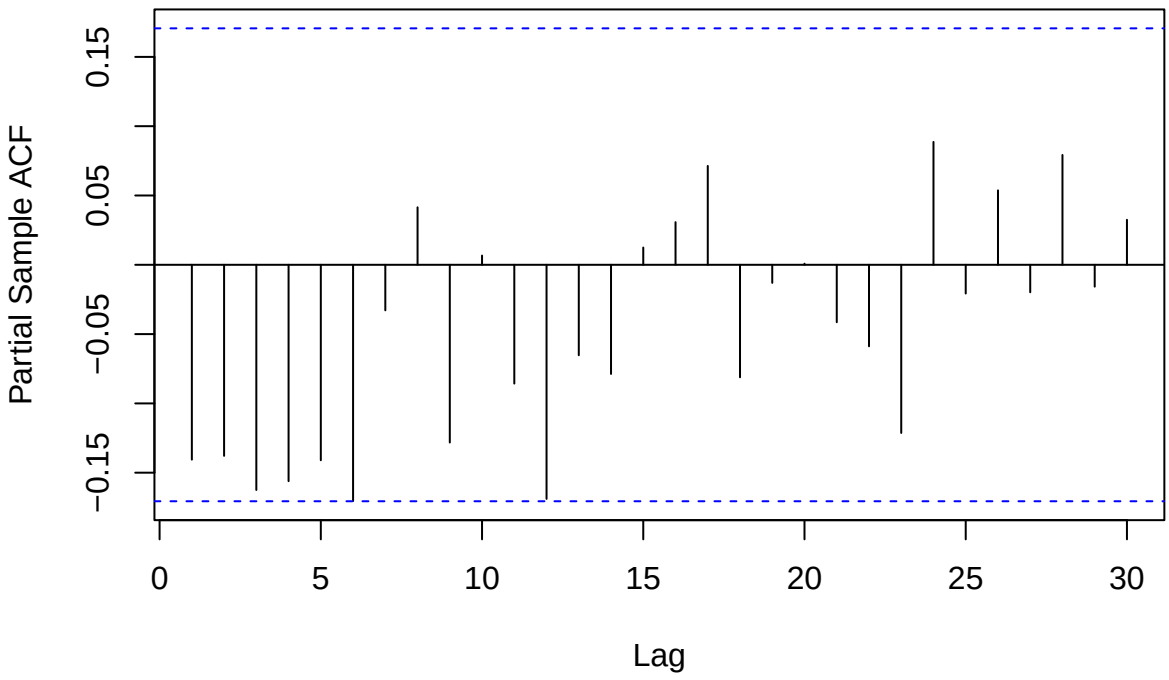
```
plot(cummeany_4, xlab="time", ylab="Mean Level First-Order Difference of VIX", type="l")
```




```
acf(y_4, lag.max = 30, na.action = na.pass, xlab="Lag", ylab="Sample ACF", main="")
```




```
pacf(y_4, lag.max = 30, na.action = na.pass, xlab="Lag", ylab="Partial Sample ACF", main="")
```




```
y_4.adf = adf.test(y_4)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -13.14    0.01
## [2,]   1  -9.71    0.01
## [3,]   2  -8.65    0.01
## [4,]   3  -7.94    0.01
## [5,]   4  -7.51    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -13.09    0.01
## [2,]   1  -9.68    0.01
## [3,]   2  -8.61    0.01
## [4,]   3  -7.91    0.01
## [5,]   4  -7.47    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -13.05    0.01
## [2,]   1  -9.64    0.01
## [3,]   2  -8.58    0.01
## [4,]   3  -7.90    0.01
## [5,]   4  -7.52    0.01
```

```
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
y_4.kpss = kpss.test(y_4)
```

```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag   stat p.value
##    2 0.0336     0.1
## -----
##  Type 2: with drift no trend
##  lag   stat p.value
##    2 0.0295     0.1
## -----
##  Type 1: with drift and trend
##  lag   stat p.value
##    2 0.0137     0.1
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

```
y_4.eacf = eacf(y_4, ar.max=5, ma.max=5)
```

```
## AR/MA
##   0 1 2 3 4 5
## 0 o o o o o o
## 1 x o o o o o
## 2 x x o o o o
## 3 x o x o o o
## 4 x x o o o o
## 5 x x o o o o
```

```
y_4.aic = matrix(0, 5, 5)
y_4.bic = matrix(0, 5, 5)

for (i in 0:4) for (j in 0:4){
  y_4.fit = arima(y_4, order=c(i, 0, j), method="ML", include.mean=TRUE)
  y_4.aic[i+1, j+1] = AIC(y_4.fit)
  y_4.bic[i+1, j+1] = BIC(y_4.fit)
}

y_4.aic_vec = sort(unmatrix((y_4.aic), byrow=FALSE))[1:13]
y_4.bic_vec = sort(unmatrix((y_4.bic), byrow=FALSE))[1:13]
```

```
y_4.aic_vec
```

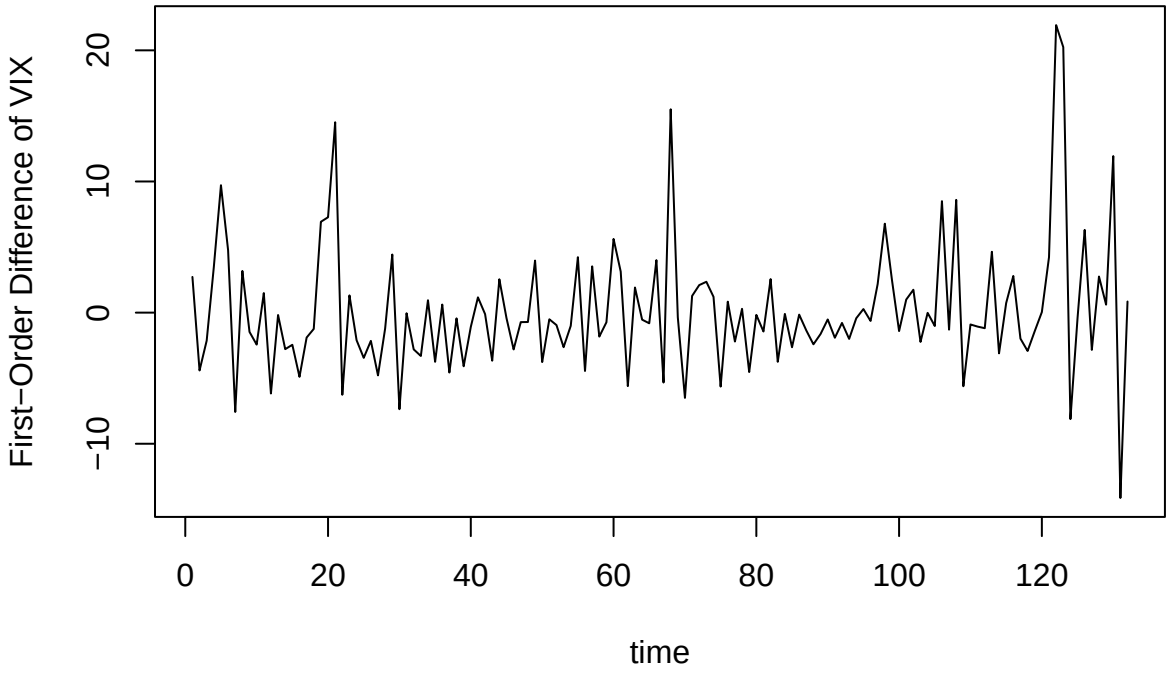
```
##    r2:c2    r3:c2    r2:c3    r1:c4    r3:c3    r4:c2    r2:c4    r1:c5
## 804.9935 806.3558 806.4781 806.8455 807.7486 807.7648 808.0079 808.0258
##    r4:c4    r3:c4    r5:c2    r4:c3    r2:c5
## 808.9517 809.6335 809.6455 809.6595 809.9428
```

```
y_4.bic_vec
```

```
##    r2:c2    r3:c2    r2:c3    r1:c4    r1:c3    r3:c3    r4:c2    r2:c4
## 816.5247 820.7698 820.8921 821.2595 822.1057 825.0454 825.0616 825.3047
```

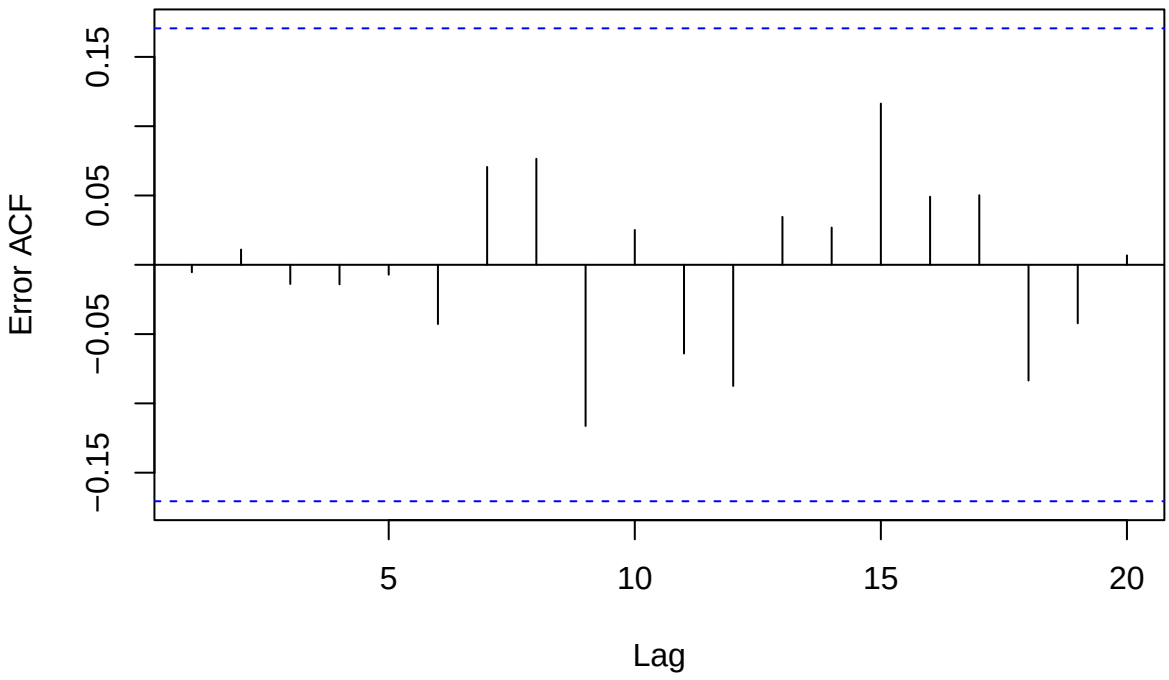
```
##    r1:c5    r1:c1    r1:c2    r2:c1    r3:c4
## 825.3226 825.4699 826.5090 827.7278 829.8131
```

```
y_4.fit1 = arima(y_4, order=c(2, 0, 2), method="ML", include.mean = FALSE)
plot(residuals(y_4.fit1), xlab="time", ylab="First-Order Difference of VIX", type="l")
```

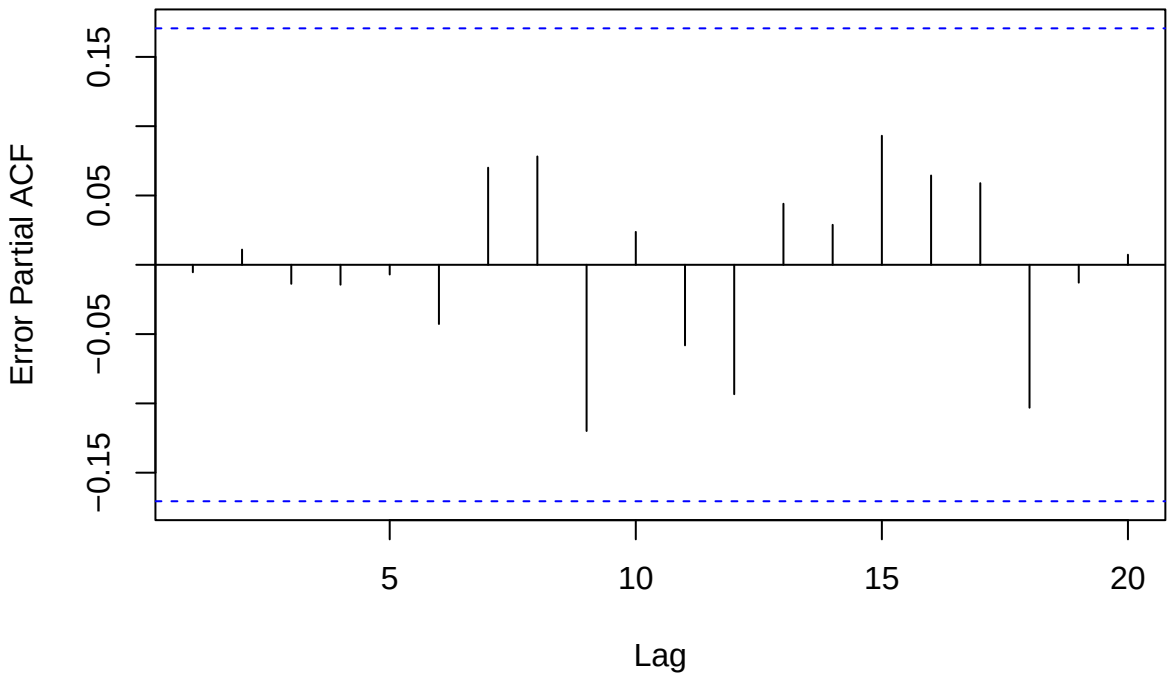


Model Diagnostics

```
acf(residuals(y_4.fit1), lag.max = 20, na.action = na.pass, xlab="Lag", ylab="Error ACF", main="")
```



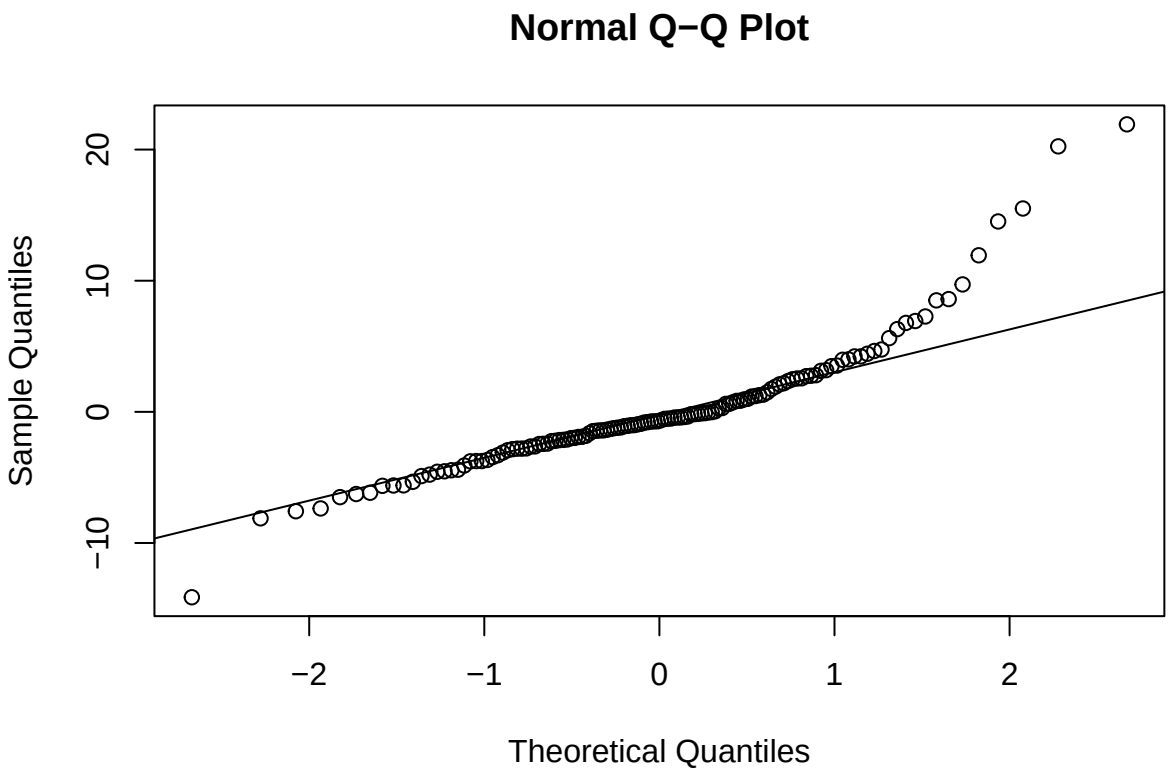

```
pacf(residuals(y_4.fit1), lag.max = 20, na.action = na.pass, xlab="Lag", ylab="Error Partial ACF", main=
```




```
Box.test(residuals(y_4.fit1), lag=1)
```

```
##
##  Box-Pierce test
##
## data:  residuals(y_4.fit1)
## X-squared = 0.0037978, df = 1, p-value = 0.9509
```

```
qqnorm(residuals(y_4.fit1))
qqline(residuals(y_4.fit1))
```




```
#hist(residuals(y_4.fit1), col='steelblue')
shapiro.test(residuals(y_4.fit1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(y_4.fit1)
## W = 0.88009, p-value = 6.38e-09
```

```
ks.test(residuals(y_4.fit1), "pnorm")
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  residuals(y_4.fit1)
## D = 0.30718, p-value = 3.036e-11
## alternative hypothesis: two-sided
```

Parameter Estimation

```
y_3.param = y_3.fit1$coef
y_4.param = y_4.fit1$coef
```

Model Specification

```
y_3.fit1
```

```
##
## Call:
## arima(x = y_3, order = c(2, 0, 2), include.mean = FALSE, method = "ML")
##
## Coefficients:
##          ar1     ar2      ma1      ma2
##       0.1451  0.1463  -0.5309  -0.1752
## s.e.  0.7774  0.2727   0.7805   0.5609
##
## sigma^2 estimated as 9.073e-05:  log likelihood = 1112.65,  aic = -2217.3
```

```
(AIC(y_3.fit1))
```

```
## [1] -2215.301
```

```
(BIC(y_3.fit1))
```

```
## [1] -2196.097
```

```
harmonic.mean(y_3.fit1$residuals)
```

```
## [1] -0.01367674
```

```
y_4.fit1
```

```
##
## Call:
## arima(x = y_4, order = c(2, 0, 2), include.mean = FALSE, method = "ML")
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.2399  -0.4602  -1.5184  0.5750
## s.e.  1.2472   0.6939   1.3057  1.1867
##
## sigma^2 estimated as 24.18:  log likelihood = -397.9,  aic = 803.8
```

```
(AIC(y_4.fit1))
```

```
## [1] 805.8008
```

```
(BIC(y_4.fit1))
```

```
## [1] 820.2148
```

```
harmonic.mean(y_4.fit1$residuals)
```

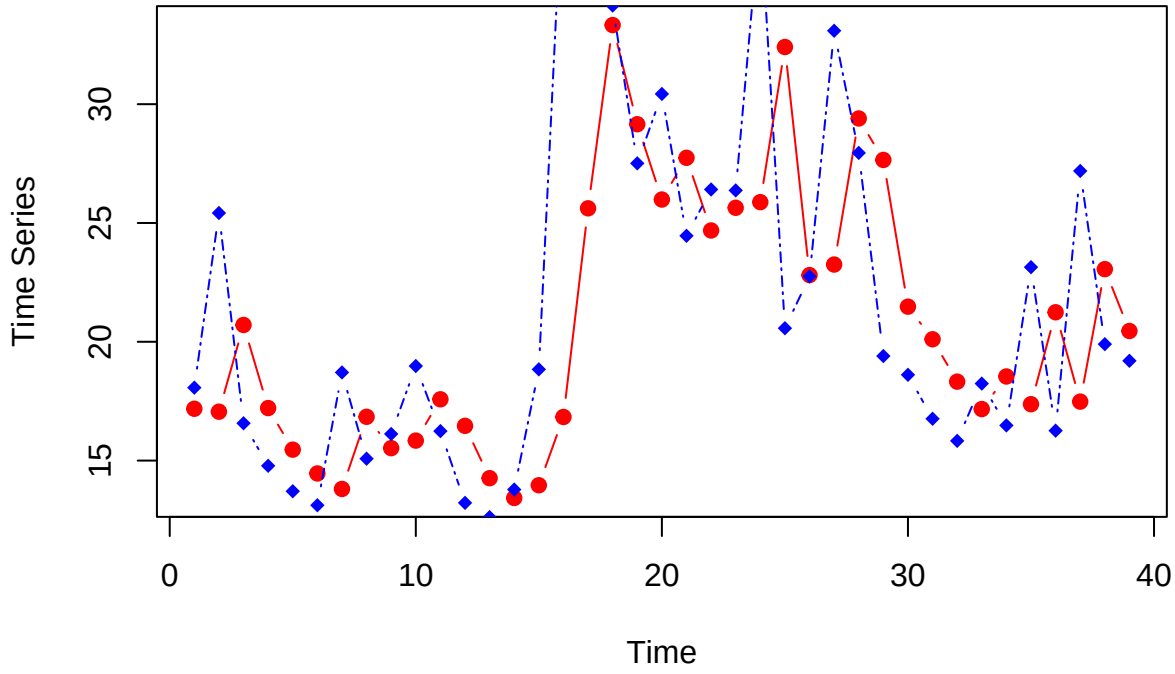
```
## [1] -1.191647
```

Model Forecasting

```
nTrain = length(train$VIX)
nTest = length(test$VIX)

test.forecast = matrix(0, nTest, 1)
test.res = matrix(0, nTest, 1)
test.se = matrix(0, nTest, 1)

for (i in (nTrain:(nTrain+nTest-1))){
```

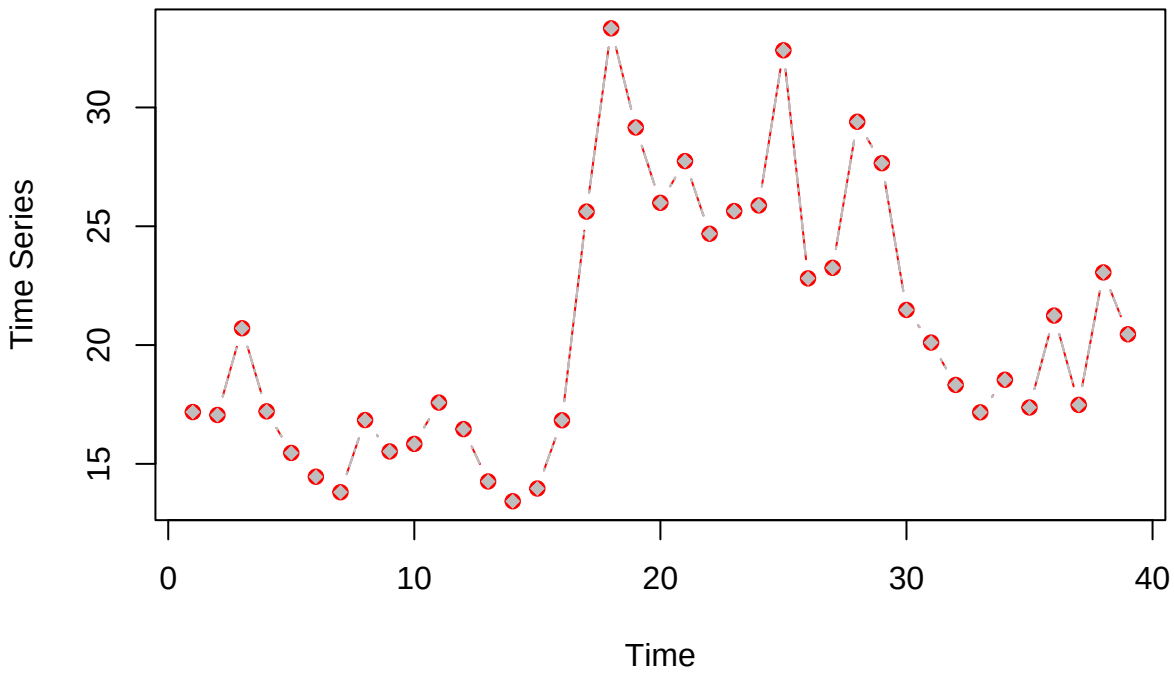
```
  test.fit = arima(1/(data$VIX[1:i]), order=c(2, 1, 1), method="ML", include.mean=TRUE)
  output = predict(test.fit)
  test.forecast[i-nTrain+1] = 1/(as.numeric(output$pred)+.Machine$double.eps)
  test.res[i-nTrain+1, 1] = 1/(as.numeric(output$pred)+.Machine$double.eps) - data$VIX[i+1]
  test.se[i-nTrain+1, 1] = as.numeric(output$se)
}
```

```
plot(unlist(test.forecast), type="b", pch=19, col="red", xlab="Time", ylab="Time Series")
lines(data$VIX[(nTrain+1):(nTrain+nTest)], pch=18, col="blue", type="b", lty=12)
```



No segmentation

```
plot(unlist(test.forecast), type="b", pch=19, col="red", xlab="Time", ylab="Time Series")
lines(unlist(test.forecast)+unlist(test.se), pch=18, col="grey", type="b", lty=12)
lines(unlist(test.forecast)-unlist(test.se), pch=18, col="grey", type="b", lty=12)
```



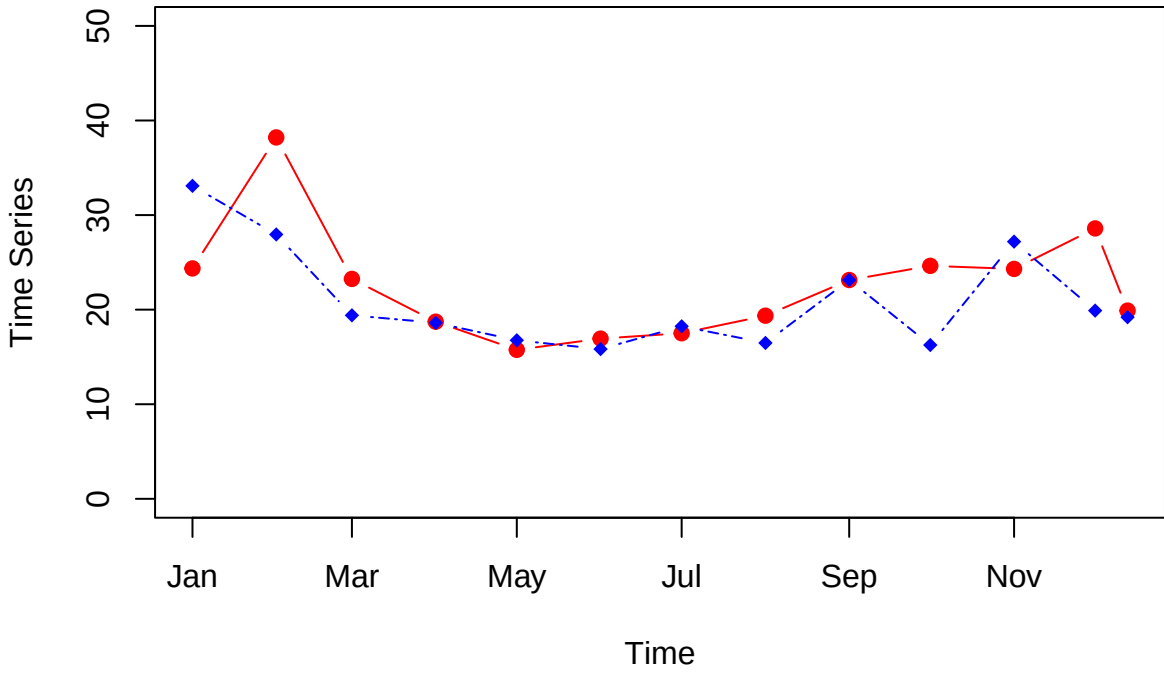

```
RMSE <- function(x){
  fval = sqrt(sum(x^2)/length(x))
  return(fval)
}

RMSE(test.res)
```

```
## [1] 7.48193
```

```
nTrain = length(train_seg$VIX)
nTest = length(test_seg$VIX)

test_seg.forecast = matrix(0, nTest, 1)
test_seg.res = matrix(0, nTest, 1)
test_seg.se = matrix(0, nTest, 1)

for (i in (nTrain:(nTrain+nTest-1))){
  test_seg.fit = arima((data_seg$VIX[seg_start:i]), order=c(2, 1, 2), method="ML", include.mean=TRUE)
  output = predict(test_seg.fit, se.fit = TRUE)
  test_seg.forecast[i-nTrain+1] = (as.numeric(output$pred))
  test_seg.res[i-nTrain+1, 1] = (as.numeric(output$pred)) - data$VIX[i+1]
  test_seg.se[i-nTrain+1, 1] = (as.numeric(output$se))
}

test_seg.forecast = data.frame(test_seg.forecast)
```

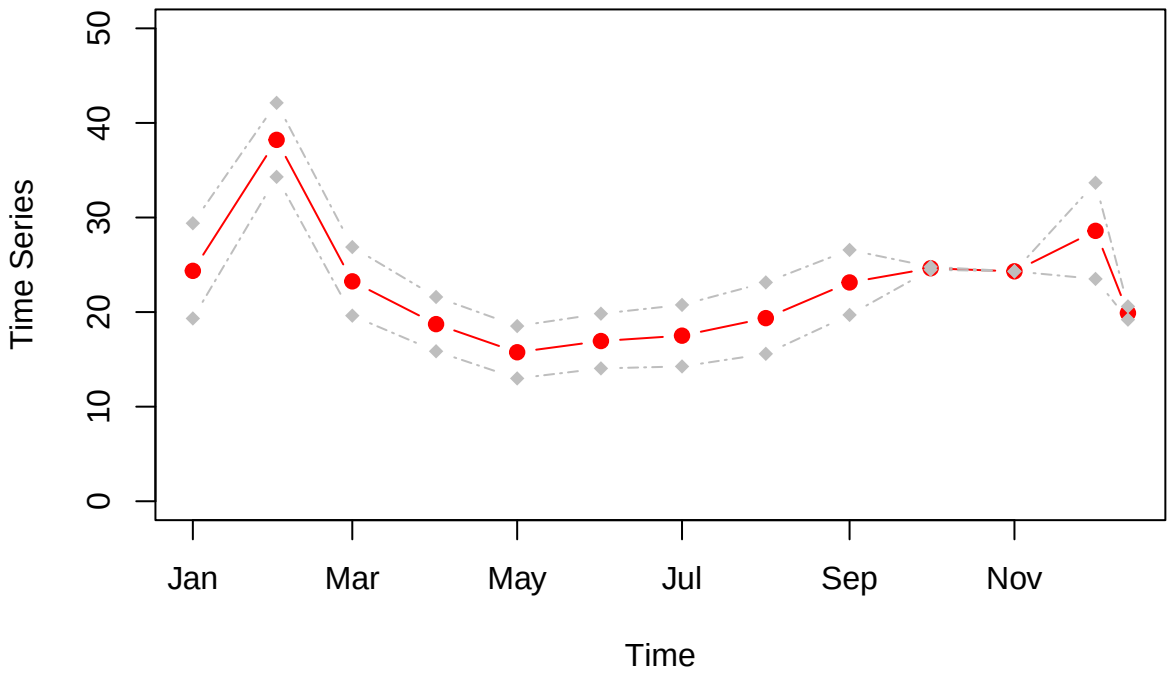
```
names(test_seg.forecast)[1] = "VIX"
test_seg.forecast$Date = test_seg$Date
test_seg.forecast = test_seg.forecast[c("Date", "VIX")]

test_seg.res = data.frame(test_seg.res)
names(test_seg.res)[1] = "VIX"
test_seg.res$Date = test_seg$Date
test_seg.res = test_seg.res[c("Date", "VIX")]

test_seg.se = data.frame(test_seg.se)
names(test_seg.se)[1] = "VIX"
test_seg.se$Date = test_seg$Date
test_seg.se = test_seg.se[c("Date", "VIX")]
```

```
plot(test_seg.forecast, type="b", pch=19, col="red", xlab="Time", ylab="Time Series", ylim=c(0,50))
lines(test_seg, pch=18, col="blue", type="b", lty=12)
```



With segmentation

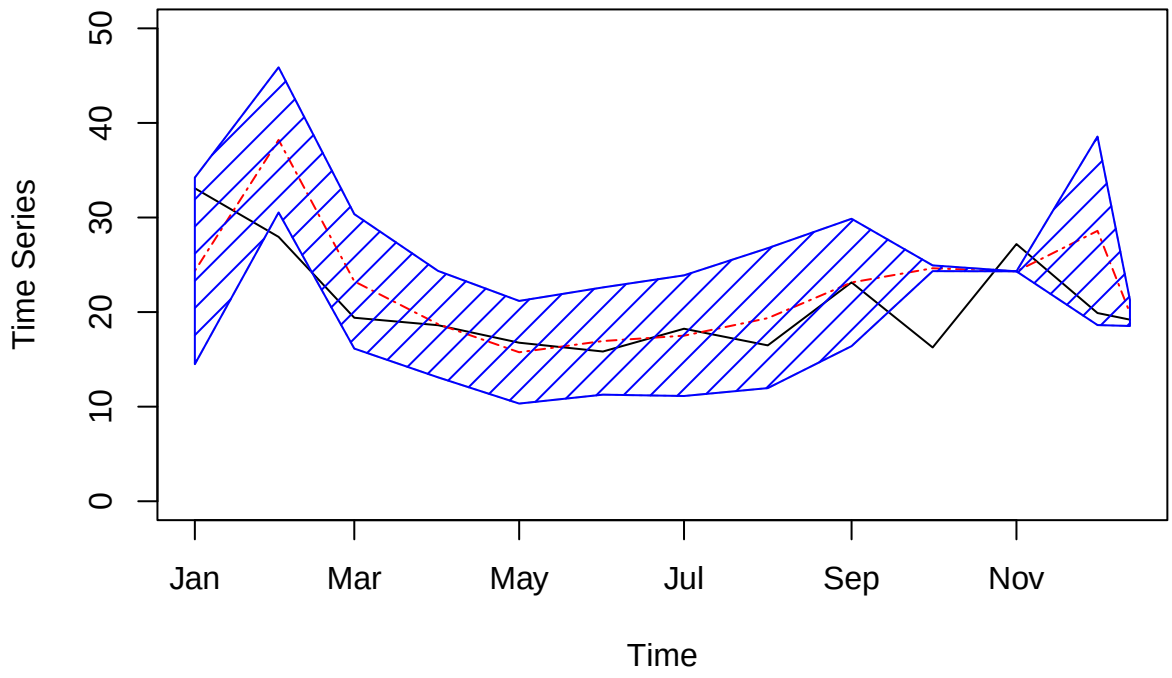
```
plot(test_seg.forecast, type="b", pch=19, col="red", xlab="Time", ylab="Time Series", ylim=c(0, 50))
lines(x=test_seg.forecast$Date, y=test_seg.forecast$VIX+test_seg.se$VIX, pch=18, col="grey", type="b", l
lines(x=test_seg.forecast$Date, y=test_seg.forecast$VIX-test_seg.se$VIX, pch=18, col="grey", type="b", l
```




```
RMSE <- function(x){
  fval = sqrt(sum(x^2)/length(x))
  return(fval)
}

RMSE(test_seg.res$VIX)
```

```
## [1] 7.622863
```

```
plot(test_seg, type="l", pch=19, col="black", xlab="Time", ylab="Time Series", ylim=c(0, 50))
lines(x=test_seg.forecast$Date, y=test_seg.forecast$VIX, pch=18, col="red", type="l", lty=12)
lines(x=test_seg.forecast$Date, y=(test_seg.forecast$VIX + 1.96*test_seg.se$VIX), pch=18, col="grey", ty
lines(x=test_seg.forecast$Date, y=(test_seg.forecast$VIX - 1.96*test_seg.se$VIX), pch=18, col="grey", ty
polygon(c(test_seg.forecast$Date, rev(test_seg.forecast$Date)), c((test_seg.forecast$VIX + 1.96*test_seg
        col = "blue", density = 10)
```




```
plot(data, type="l", col="black")
lines(test_seg.forecast, type="l", col="red")
lines(x=test_seg.forecast$Date, y=(test_seg.forecast$VIX + 1.96*test_seg.se$VIX), pch=18, col="grey", ty
lines(x=test_seg.forecast$Date, y=(test_seg.forecast$VIX - 1.96*test_seg.se$VIX), pch=18, col="grey", ty
polygon(c(test_seg.forecast$Date, rev(test_seg.forecast$Date)), c((test_seg.forecast$VIX + 1.96*test_seg
        col = "blue", density = 10)
```